

A spatiotemporal model with visual attention for video classification

Mo Shan and Nikolay Atanasov

Department of Electrical and Computer Engineering
University of California San Diego, La Jolla, California, USA
Email: {moshan, natanasov}@eng.ucsd.edu

Abstract—High level understanding of sequential visual input is important for safe and stable autonomy, especially in localization and object detection. While traditional object classification and tracking approaches are specifically designed to handle variations in rotation and scale, current state-of-the-art approaches based on deep learning achieve better performance. This paper focuses on developing a spatiotemporal model to handle videos containing moving objects with rotation and scale changes. Built on models that combine Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to classify sequential data, this work investigates the effectiveness of incorporating attention modules in the CNN stage for video classification. The superiority of the proposed spatiotemporal model is demonstrated on the Moving MNIST dataset augmented with rotation and scaling.

I. INTRODUCTION

Semantic understanding of sequential visual input is vital for autonomous robots to perform localization and object detection. Moreover, robust models have to be adaptive to scenes containing multiple interacting objects. For instance, self-driving cars need to classify scenes that contains small and large vehicles for loop closure. Similarly, robots have to infer the classes of objects in order to grasp them, while the objects could be rotated in different ways. Although videos contain richer information than individual images, how to effectively harness the temporal dependency in sequential visual input remains an open problem. Particularly, this paper investigates video classification in the presence of moving objects with rotation and scale changes. Preliminary results on digit gesture classification demonstrate the extensibility of the proposed model on handling articulated objects such as hand.

Problem Formulation. Given a training dataset that contains videos $D_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}_{>0}^{w \times h \times c \times t}$ is the i_{th} video, w, h, c is the size for each frame, t is the number of frames, and $y_i \in \{1, 2, \dots, K\}$ is the i_{th} class label, we aim to train a classifier C for unseen data $(x_i, y_i) \in D_{test}$, such that $C(x_i) = y_i$. For instance, in the context of action recognition, each video x_i is labeled with a class label y_i such as walking or running.

Initial work [9] approaches this task using CNNs, but the majority of recent works such as [16] rely more on RNNs that are adept at processing sequential data. While RNNs are good at dealing with contextual information from the past, CNNs are useful to extract hierarchical features from video frames. Motivated by the advantages of both types of networks, this

paper proposes a spatiotemporal model in which CNN and RNN are concatenated, as shown in Fig. 1.

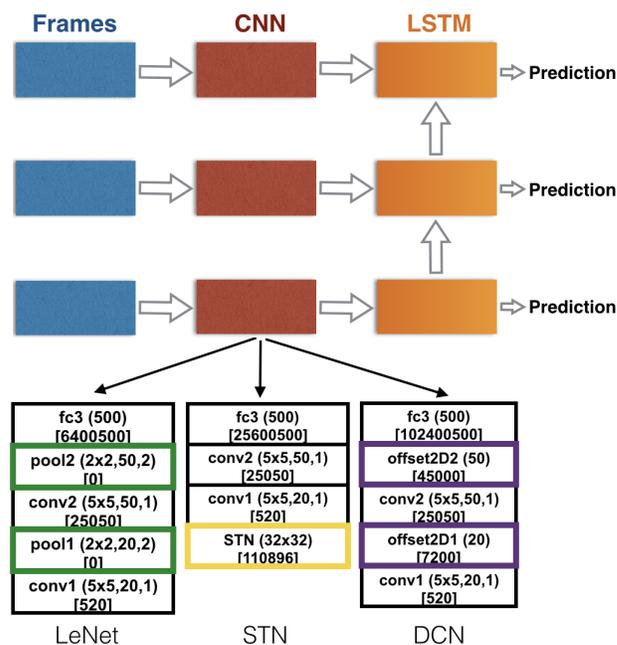


Fig. 1: Architecture of the proposed model: The spatiotemporal model contains a CNN for feature extraction and a LSTM for temporal processing. Three CNN models are illustrated in the figure, including the baseline LeNet, and two variants of LeNet augmented with attention modules, one with STN and the other with DCN. The layers that handle geometric transformations are highlighted in the colored boxes. For the CNNs, conv is convolution layer, fc is fully-connected layer, pool is max-pooling layer. The numbers inside parentheses are: kernel size, number of output channels, and stride for conv and pool layers; number of output channels for offset2D layers; number of outputs for fc layers; size of output feature map for STN. The numbers inside the square brackets are trainable parameters in each layer given a 64×64 input image.

Despite the impressive performance of deep neural networks, previous models have fixed network depth and convolutional kernel size, which may lead to brittle performance when dealing with large and rapid variations in scale and rotation of objects in high speed robot navigation. In the proposed

spatiotemporal model, an attention module is introduced in the network architecture. This is the key idea that increases the robustness of the model to geometric transformations such as object rotation and scale changes commonly seen in the real world. Previously, networks with visual attention were developed to limit the amount of information that needs to be processed and increase their robustness against cluttered background. However, these attention modules have been only used for single image recognition and classification tasks. This work applies attention mechanism for video classification.

Attention modules can be categorized into hard attention [14] and soft attention [8, 13, 3]. This work focuses on the latter since the former is stochastic and non-differentiable. The soft attention modules considered in the proposed spatiotemporal model include Spatial Transformer Networks (STN) [8] and Deformable Convolutional Networks (DCN) [3]. STN performs affine transformation on the feature map globally, whereas DCN samples the feature map locally and densely via learnt offsets to deform the convolutional kernel. LeNet [12] which has a simple structure and no attention module is used as baseline for comparison.

To summarize, the novelty of the paper is a spatiotemporal model with visual attention tailored for video classification, which enables robustness to multiple objects with rotation and scale changes.

II. SPATIAL TEMPORAL MODEL WITH VISUAL ATTENTION

A. Feature extraction

Given a sequence of images, hierarchical features are extracted using a model similar to LeNet. The lower layers of LeNet include two convolution and max-pooling blocks. Its upper layers contain a fully-connected layer, and a softmax layer used for classification during pre-training, but removed for feature extraction. Although the model is simple, it still contains the essence of deeper CNNs. The trainable parameters are the weights and bias terms in the convolutional and fully-connected layers shown in Fig. 1. The proposed model uses an attention module prior to the first convolution layer instead of max-pooling layers to deal with geometric transformation, which means max-pooling layers are removed and the remaining CNN layers are augmented by attention modules. There are two variations of the structure, one with STN and the other with DCN.

STN consists of three components: localization network, grid generator, and bilinear sampler as depicted in Fig. 2. The localization network consists of 2 max-pooling layers, 2 convolution layers, and 2 fully-connected layers. It takes the image \mathcal{I} as input and learns the affine transformation A_θ that should be applied to the image, i.e. $\theta = f_{loc}(\mathcal{I})$, where θ represents the transformation parameters. The grid generator produces coordinates $A_\theta(\mathcal{G})$ based on A_θ and a regular grid \mathcal{G} , which are the sampling locations in the input image to produce the output feature map. The bilinear sampler takes input image and $A_\theta(\mathcal{G})$ to generate the output feature map using bilinear interpolation. Because of the affine transformation, STN is trained to focus on the digits via cropping, translation and

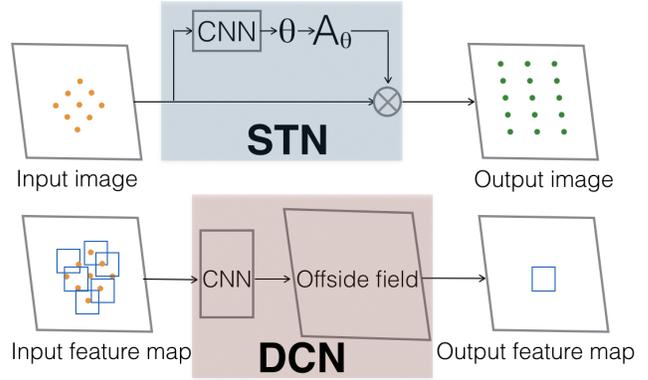


Fig. 2: Structure of STN and DCN.

isotropic scaling of the input image. The output size of STN is 32×32 . Different output sizes of STN are tested, but that does not affect the classification accuracy significantly. Since additional layers are introduced in the localization network, there are more trainable parameters in the proposed model as described in Fig. 1.

DCN abandons the parametric transformation adopted by STN and its variants. As shown in Fig. 2, the spatial sampling locations in the convolution kernels are augmented with additional offsets learnt from classification. For example, a 5×5 kernel with dilation 1 in a convolutional layer samples on grid $\mathcal{R} = \{(-2, -2), (-2, -1), \dots, (1, 2), (2, 2)\}$ on input feature map x . The feature at p_0 in output feature map y could be obtained based on Eq. (1a). The grid \mathcal{R} is augmented by offsets $\{\Delta p_n | n = 1, \dots, N\}$, $N = |\mathcal{R}|$ in deformable convolution as in Eq. (1b), where the offset is implemented by bilinear interpolation. Δp_n is learnt via applying a convolutional layer over x , which has the same kernel size of 5×5 , and same spatial resolution with x . The output of the offset fields has $2N$ channels, which encodes N vectors for Δp_n in $2D$ denoted as offset2D layers in Fig. 1. The trainable parameters include the offsets $\{\Delta p_n | n = 1, \dots, N\}$ as well as the convolutional kernels.

$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n) \quad (1a)$$

$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \quad (1b)$$

B. Temporal processing

The aforementioned CNN layers are connected to Long Short Term Memory (LSTM) [7] with a softmax layer to classify each frame as depicted in Fig. 1. Including LSTM breaks the dependence on the size of network, because a much deeper CNN would be required to replace the RNN for video classification. Note that the dimension of the output space of LSTM is 8 only and there is no dropout as the focus is on comparing the CNN layers. The CNN processes every temporal slice of the video independently with the weights fixed. In contrast, the LSTM has memory blocks to learn long

term dependencies in the video since the output is dependent on the previous hidden state.

As shown in Eq. (2), LSTM contains a cell state c_t , which is controlled by input gate i_t , forget gate f_t , and output gate o_t . f_t decides which information to throw away, i_t determines what to store in c_t , while o_t generates the output based on the input x_t as well as the previous hidden state h_{t-1} . The W, b are the trainable parameters.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2a)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2b)$$

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \quad (2c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (2d)$$

III. DATASET CREATION

A practical challenge for training CNNs and RNNs for video classification with visual attention is the lack of datasets that exhibit sufficient variation in rotation and scale, because existing video classification benchmarks such as UCF101 [19] and Sports-1M [9] are collected from daily activities. To address this issue, this work proposes a synthetic dataset based on Moving MNIST [20], with augmentation that includes rotation and scaling. This dataset is also overfit-resistant as a side benefit of applying augmentation, as analyzed in [21].

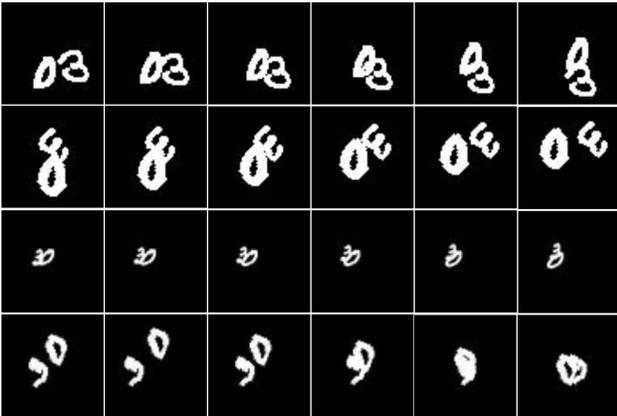


Fig. 3: A sequence of images in class 03 in augmented Moving MNIST dataset. From first row to last row: original images, rotated images, scaled images, rotated and scaled images.

In our dataset, D_{train} and D_{test} that contain MNIST digits are created. Two samples $(\mathcal{I}_m, d_m), (\mathcal{I}_n, d_n) \in \mathcal{M}$ are randomly chosen from the MNIST training set, where $\mathcal{I}_m, \mathcal{I}_n$ are the images and $d_m, d_n \in \{0, 1, \dots, 9\}$ are class labels. $\mathcal{I}_m, \mathcal{I}_n$ are placed at two random locations on a canvas \mathcal{H} of size 64×64 . They are assigned with a velocity in range $[4, 12]$ whose direction is chosen at random. $\mathcal{I}_m, \mathcal{I}_n$ bounce off the boundary and could overlap. Furthermore, each canvas \mathcal{H} is augmented with controllable variations of rotation and scaling to simulate the scene that contains multiple interacting objects, i.e. random rotation $\mathcal{R} \in [0, 2\pi]$ and scaling $\mathcal{S} \in [0.4, 1]$ are applied to obtain $\hat{\mathcal{H}}$. Several sequences of $\hat{\mathcal{H}}$ are used

to produce the video x_i , and the labels for two digits are concatenated as the new class label $y_i = d_m d_n$. Sample images in the dataset are displayed in Fig. 3. Essentially, the video classification in this context is to recognize the digits in the presence of transformation and occlusion.

IV. EXPERIMENT

A. Training details

The networks are implemented based on [1, 11, 6] using Keras [2]. During pre-processing, the images are centered and converted to range $[0, 1]$. The models are pre-trained on the Scaled MNIST dataset, which is the MNIST dataset augmented by translation and scaling. The maximum random horizontal and vertical shifts in terms of fraction of total width and height is 0.2. Range for random zoom is $[1.0, 2.5]$. The kernel weights matrices are initialized via Xavier uniform initializer, and the bias terms are initialized to 0. The objective function is the cross entropy loss, and the optimizer for training is Adadelta optimizer with a learning rate of 1. The cross entropy loss and accuracy are 0.054, 98.31% for LeNet, 0.026, 99.14% for STN, and 0.072, 97.62% for DCN.

There are 100 classes in augmented Moving MNIST dataset, and 10 different rotations or scales in each class, with 5 frames for one setting. Hence there are 5000 images for training and 5000 images for testing. The training is conducted in 10 epochs with RMSProp optimizer using a learning rate of 1×10^{-3} and a batch size of 50. The objective function is categorical cross entropy loss. The kernel weights matrix in LSTM is initialized via Xavier uniform initializer, and the recurrent kernel weights matrix is initialized by a random orthogonal matrix. The bias vector is initialized to 0. Each time the LSTM is trained for 300 rounds and the loss and accuracy are averaged to reduce the variance.

Note that the CNN and RNN are trained separately because STN and DCN introduce much more trainable parameters as shown in Fig. 1. This issue is more severe for DCN as the number of trainable offsets increases linearly with respect to the size of the input feature map. For instance, there are 102,483,280 trainable parameters in total for DCN given a 64×64 image, while only 6,431,080 parameters for LeNet, meaning that DCN has 16 times more parameters than LeNet. As a result, the 12GB GPU memory used for the experiments becomes insufficient once the attention modules are distributed along the time dimension, making it difficult to train CNN and RNN together.

B. Quantitative results on augmented Moving MNIST

The cross entropy loss and test accuracy are shown in Table I. The proposed model STN-LSTM and DCN-LSTM are compared against LeNet-LSTM as baseline. As for video classification, DCN-LSTM consistently performs better than LeNet-LSTM, as the former has lower cross entropy loss and higher accuracy in all cases. This shows that although max-pooling layers could deal with rotation or scaling to some extent, DCN is superior than the max-pooling layers at handling deformations. Moreover, Table I also shows that

TABLE I: Comparison of cross entropy loss and test accuracy for the proposed model and baseline.

Moving MNIST	LeNet-LSTM	STN-LSTM	DCN-LSTM
Normal	1.44, 97.96%	1.98, 87.26%	1.27, 99.62%
Rotation	1.42, 98.43%	1.97, 90.47%	1.29, 99.70%
Scaling	1.52, 96.28%	1.99, 86.90%	1.28, 99.41%
Rotation+Scaling	1.51, 96.82%	1.99, 89.10%	1.25, 99.46%

STN-LSTM performs poorly compared to LeNet-LSTM and DCN-LSTM. Since there are two digits in each image and STN performs affine transformation globally, it is hard to attend to the digits individually, resulting in the low accuracy.

Regarding different transformation, DCN-LSTM is hardly affected no matter which kind of augmentation is applied, as the accuracy stays above 99%. The variance in the accuracy of LeNet-LSTM is also small, indicating that max-pooling layers achieve similar results for different augmentation. In contrast, STN-LSTM handles rotated images significantly better than scaled images. Additionally, all models struggle with scaled images, as the accuracy in the third row is lowest. Furthermore, scaling is more challenging than rotation as the architectures could reach a relatively higher accuracy for rotated images.

C. Qualitative analysis on augmented Moving MNIST

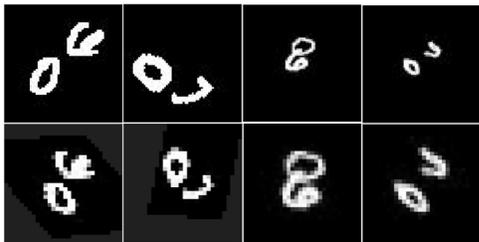


Fig. 4: Output images of STN for class 06. First row: input images. Second row: output images of STN. From first column to last column: original images, rotated images, scaled images, rotated and scaled images.

This section presents a qualitative analysis to shed some light on why STN does not do well. From the images processed by STN shown in Fig. 4, it could be observed that all the images are centered and transformed to a canonical orientation. There are several reasons when this kind of processing could be disadvantageous. For example, the original image is zoomed out when the digits are far from each other, even though doing so would make the digits smaller. This may be the reason why STN-LSTM has lower accuracy for images without any augmentation as shown in Table I. In addition, STN rotates the two digits in the same way in the rotated image. This may be helpful when the same rotation is applied to both digits as what is done in the dataset, but not so when digits are rotated differently. To summarize, the inability of one STN module to attend to each digit separately affects its classification performance negatively. To address this issue,

STN could be connected with RNN to detect multiple digits as proposed in [18]. The inference framework proposed in [4] could be used as well to attend to scene elements one at a time.

D. Classification of digit gesture

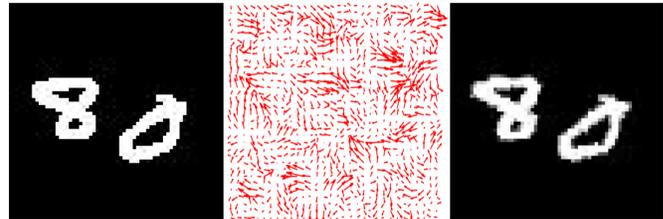


Fig. 5: Image distorted by elastic deformation. From left to right: original image, elastic deformation field, deformed image.

Digit gesture has been widely used for evaluating the recognition algorithms, and classification of digit gesture is more challenging when the digits are written in the air by hand [10]. In this experiment, elastic deformation described in [17] is applied to the images in Moving MNIST dataset, with rotation and scaling augmentation, to simulate oscillations of hand muscles. Random displacement fields $(\Delta x, \Delta y)$ are convolved with a Gaussian filter with standard deviation $\sigma = 10$, and size $[7, 7]$, then multiplied by a scaling factor $\alpha = 300$. The resulting images are similar to those in the TVC-hand gesture database [10], which contains images generated by accumulating the center points of a hand over time. A sample image after applying elastic deformation is shown in Fig. 5.

The cross entropy loss and accuracy are 1.48, 97.19% for LeNet-LSTM, 1.48, 97.19% for STN-LSTM, and 1.28, 99.30% for DCN-LSTM, indicating that DCN-LSTM is still the best model to deal with distorted digits because of its deformable convolutional layers. [5] shows that Deformable Parts Models (DPM) could be formulated as CNNs. Similarly, deformable convolutional layers in the DCN-LSTM could be modified to explicitly learn the deformation field, which is better than DPM for certain classes [15]. Therefore, the capability of the proposed model in recognizing digit gesture suggests that the model could be extended for tracking hand trajectory or dealing with multi-part articulated objects such as hand.

V. DISCUSSION

This paper describes a spatiotemporal model for video classification with the presence of rotated and scaled objects. The effectiveness of DCN-LSTM over the baseline is demonstrated using the augmented Moving MNIST dataset. Moreover, preliminary results on digit gesture recognition are also shown. Regarding future work, how to train the network end to end is worth exploring. It would be more elegant to train the two stages simultaneously in an efficient way.

REFERENCES

- [1] Octavio Arriaga. Spatial transformer networks, 2017. URL https://github.com/oarriaga/spatial_transformer_networks.
- [2] François Chollet et al. Keras, 2015. URL <https://github.com/fchollet/keras>.
- [3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017.
- [4] SM Eslami, N Heess, and T Weber. Attend, infer, repeat: Fast scene understanding with generative models. arxiv preprint arxiv:..., 2016. URL <http://arxiv.org/abs/1603.08575>.
- [5] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446, 2015.
- [6] Matt Harvey. Five video classification methods implemented in keras and tensorflow, 2017. URL <https://github.com/harvitronix/five-video-classification-methods>.
- [7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [8] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
- [9] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
- [10] Seongwan Kim, Yuseok Ban, and Sangyoun Lee. Tracking and classification of in-air hand gesture based on thermal guided joint filter. *Sensors*, 17(1):166, 2017.
- [11] Felix Lau. Notes on deformable convolutional networks, 2017. URL <https://github.com/felixlaumon/deform-conv>.
- [12] Yann LeCun et al. Lenet-5, convolutional neural networks.
- [13] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. *arXiv preprint arXiv:1612.03897*, 2016.
- [14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.
- [15] Marco Pedersoli, Radu Timofte, Tinne Tuytelaars, and Luc Van Gool. An elastic deformation field model for object detection and tracking. *International Journal of Computer Vision*, 111(2):137–152, January 2015.
- [16] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.
- [17] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.
- [18] Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*, 2015.
- [19] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [20] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015.
- [21] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.