Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation

—

Supplementary Material

Anurag Ranjan[1] Varun Jampani[2] Lukas Balles[1]
Kihwan Kim [2] Deqing Sun [2] Jonas Wulff [1,3] Michael J. Black[1]
[1]Max Planck Institute for Intelligent Systems [2]NVIDIA [3]MIT
{aranjan, lballes, jwulff, black}@tuebingen.mpg.de
{vjampani, kihwank, deqings}@nvidia.com

A. Appendix

A.1. Competitive Collaboration as a General Learning Framework

Competitive collaboration (CC) can be seen as a general learning framework for training multiple task-specific networks. To showcase this generality, we demonstrate CC on a mixed-domain classification problem in Section A.1.1 and analyze CC convergence properties in Section A.1.2.

A.1.1 Mixed Domain Classification

Digit classification is the task of classifying a given image I into one of the 10 digit classes $t \in \{0, 1, 2, .., 9\}$. Two most widely used datasets for digit classification include images of the postal code digits, MNIST [6] and street view house numbers, SVHN [9]. For our setup, we take the samples from both of the datasets, and shuffle them together. This means that, although an image and a target, (I_i, t_i) form a pair, there is no information if the digits came from MNIST or SVHN.

We now train our model under Competitive Collaboration framework given the mixed-domain dataset MNIST+SVHN, a mixture of MNIST and SVHN. The model consists of two networks R_x and F_x that compete with each other regulated by a moderator M_y which assigns training data to each of the competitors. Here, x denotes the combined weights of the two competitor networks (R, F) and y denotes the weight of the moderator network M. The networks are trained using an alternate optimization procedure consisting of two phases. In the competition phase, we train the competitors by fixing the moderator M and minimizing,

$$E_1 = \sum_i m_i \cdot H(R_x(I_i), t_i) + (1 - m_i) \cdot H(F_x(I_i), t_i) \quad (1)$$

where $m_i = M_y(I_i) \in [0, 1]$ is the output of the moderator and is the probability of assigning a sample to R_x. $H(R_x(I_i), t_i)$ is the cross entropy classification loss on the network R_x and a similar loss is applied on network F_x.

During the collaboration phase, we fix the competitors and train the moderator by minimizing,

$$E_2 = E_1 + \sum_i \lambda \cdot \begin{cases} -\log(m_i + \varepsilon) & \text{if } L_{R_i} < L_{F_i}, \\ -\log(1 - m_i + \varepsilon) & \text{if } L_{R_i} \geq L_{F_i}. \end{cases} \quad (2)$$

where $L_{R_i} = H(R_x(I_i), t_i)$ is the cross entropy loss from network R_x and similarly $L_{F_i} = H(F_x(I_i), t_i)$. In addition to the above loss function E_1, we use an additional constraint on the moderator output that encourages the variance of m, $\sigma_m^2 = \Sigma_i (m_i - \bar{m})^2$ to be high, where $\bar{m}$ is the mean of m within a batch. This encourages the moderator to assign images to both the models, instead of always assigning them to a single model.

In an ideal case, we expect the moderator to correctly classify MNIST digits from SVHN digits. This would enable each of the competitors to specialize on either MNIST or SVHN, but not both. In such a case, the accuracy of the model under CC would be better than training a single network on the MNIST+SVHN mixture.

Experimental Results For simplicity, we use a CNN with 2 convolutional layers followed by 2 fully-connected layers for both the digit classification networks (R, F) as well as the moderator network M. Each of the convolutional layers use a kerel size of 5 and 40 feature maps. Each of the fully connected layers have 40 neurons.

We now compare the performance of the CC model on MNIST+SVHN mixture with training a single network on

	Training	M	S	M+S
R	Basic	1.34	11.88	8.96
R	CC	1.41	**11.55**	8.74
F	CC	**1.24**	11.75	8.84
R, F, M	CC	**1.24**	**11.55**	**8.70**

Table 1: Percentage classification errors. M and S refer to MNIST and SVHN respectively.

	MNIST	SVHN
R	0%	100%
F	100%	0%

Table 2: Assignments of moderator to each of the competitors.

the same dataset. We see that our performance is better on the mixture dataset as well as individual datasets (see Table 1). As shown in Table 1, the network R specializes on SVHN digits and network F specializes on MNIST digits. By using the networks (R, F, M), we get the best results as M picks the specialized networks depending on the data sample.

We also examine the classification accuracy of the moderator on MNIST and SVHN digits. We observe that moderator can accurately classify the digits into either MNIST or SVHN without any labels (see Table 2). The moderator learns to assign 100% of MNIST digits to F and 100% of SVHN digits to R. This experiment provides further evidence to support the notion that CC can be generalized to other problems.

A.1.2 Theoretical Analysis

Competitive Collaboration is an alternating optimization procedure. In the competition phase, we minimize E_1 with respect to x; in the collaboration phase we minimize $E_2 = E_1 + \lambda L_M$ with respect to y. One might rightfully worry about the convergence properties of such a procedure, where we optimize different objectives in the alternating steps.

It is important to note that—while E_1 and E_2 are different functions—they are in fact closely related. For example, they have the same minimizer with respect to the moderator output, namely assigning all the mass to the network with lower loss. Ideally, we would want to use E_1 as the objective function in both phases, but resort to using E_2 in the collaboration phase, since it has empirically proven to be more efficient in pushing the moderator towards this optimal choice.

Hence, while we are minimizing different objective functions in the competition and collaboration phases, they are closely related and have the same "goal". In the following, we formalize this mathematically by identifying general assumptions on how "similar" two functions have to be for such an alternating optimization procedure to converge. Roughly speaking, we need the gradients of the two objectives to form an acute angle and to be of similar scales. We will then discuss to what extent these assumptions are satisfied in the case of Competitive Collaboration. Proofs are outsourced to the end of this section for readability.

General Convergence Theorem Assume we have two functions

$$f, g \colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \tag{3}$$

and are performing alternating gradient descent updates of the form

$$x_{t+1} = x_t - \alpha \nabla_x f(x_t, y_t), \tag{4}$$

$$y_{t+1} = y_t - \beta \nabla_y g(x_{t+1}, y_t). \tag{5}$$

We consider the case of single alternating gradient descent for convenience in the analysis. With minor modifications, the following analysis also extends to the case of multiple gradient descent updates (or even exact minimization) in each of the alternating steps. The following Theorem formulates assumptions on f and g under which such an alternating optimization procedure converges to a first-order stationary point of f.

Theorem 1. *Assume f is lower-bounded and $x \mapsto \nabla_x f(x, y)$ is Lipschitz continuous with constant G_1 for every y and $y \mapsto \nabla_y f(x, y)$ is Lipschitz continuous with constant $G_2(x)$. Assume $\alpha \leq 2L_1^{-1}$. If there is a constant $B > 0$ such that*

$$\beta \langle \nabla_y f(x, y), \nabla_y g(x, y) \rangle \geq \frac{G_2(x)\beta^2}{2} \|\nabla_y g(x, y)\|^2 + B \|\nabla_y f(x, y)\|^2 \tag{6}$$

then (x_t, y_t) converges to a first-order stationary point of f.

Eq. (6) is a somewhat technical assumption that lower-bounds the inner product of the two gradients in terms of their norms and, thus, encodes that these gradients have to form an acute angle and be of similar scales.

Convergence of Competitive Collaboration We now discuss to what extent the assumptions for Theorem 1 are satisfied in the case of Competitive Collaboration. For the mathematical considerations to follow, we introduce a slightly more abstract notation for the objective functions of Competitive Collaboration. For a *single data point*, E_1 has the form

$$f(x, y) = M(y) L_R(x) + (1 - M(y)) L_F(x), \tag{7}$$

where $M(y) \in [0,1]$ is a function of y (the weights of the moderator) and $L_R(x), L_F(x) > 0$ are functions of x (the weights of the two competing networks). The loss function E_2 reads

$$g(x,y) = f(x,y) + \lambda \cdot \begin{cases} -\log(M(y)+\varepsilon) & \text{if } L_R(x) < L_F(x), \\ -\log(1-M(y)+\varepsilon) & \text{if } L_R(x) \geq L_F(x). \end{cases} \quad (8)$$

The following Proposition shows that f and g satisfy the conditions of Theorem 1 under certain assumptions.

Proposition 1. *Let f and g be defined by Equations (7) and (8), respectively. If $M(y)$, $L_R(x)$ and $L_F(x)$ are Lipschitz smooth, then f and g fulfill the assumptions of Theorem 1.*

The smoothness conditions on $M(y)$, $L_R(x)$, $L_F(x)$ are standard as they are, for example, needed to guarantee convergence of gradient descent for optimizing any of these objective functions individually.

This Proposition shows that the objectives for individual data points satisfy Theorem 1. In practice, however, we are concerned with multiple data points and objectives of the form

$$f(x,y) = \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(x,y), \quad (9)$$

where

$$f^{(i)}(x,y) = M^{(i)}(y)L_R^{(i)}(x) + (1-M^{(i)}(y))L_F^{(i)}(x), \quad (10)$$

and

$$g(x,y) = \frac{1}{n}\sum_{i=1}^{n} g^{(i)}(x,y), \quad (11)$$

where

$$g^{(i)}(x,y) = f^{(i)}(x,y) + \lambda \cdot \begin{cases} -\log(M^{(i)}(y)+\varepsilon) & \text{if } L_R^{(i)}(x) < L_F^{(i)}(x), \\ -\log(1-M^{(i)}(y)+\varepsilon) & \text{if } L_R^{(i)}(x) \geq L_F^{(i)}(x). \end{cases} \quad (12)$$

While we have just found a suitable lower bound on the inner product of $\nabla_y f^{(i)}$ and $\nabla_y g^{(i)}$, unfortunately, the sum structure of $\nabla_y f$ and $\nabla_y g$ makes it really hard to say anything definitive about the value of their inner product. It is *plausible* to assume that $\nabla_y f$ and $\nabla_y g$ will be sufficiently close to guarantee convergence in practical settings. However, the theory developed in Theorem 1 does not directly apply.

A.1.3 Proofs

Proof of Theorem 1. The update of x is a straight-forward gradient descent step on f. Using the Lipschitz bound on f, we get

$$\begin{aligned} f(x_{t+1},y_t) &\leq f(x_t,y_t) - \alpha\langle \nabla_x f(x_t,y_t), \nabla_x f(x_t,y_t)\rangle \\ &\quad + \frac{G_1\alpha^2}{2}\|\nabla_x f(x_t,y_t)\|^2 \\ &= f(x_t,y_t) - \left(\alpha - \frac{G_1\alpha^2}{2}\right)\|\nabla_x f(x_t,y_t)\|^2 \\ &\leq f(x_t,y_t) - A\|\nabla_x f(x_t,y_t)\|^2 \end{aligned} \quad (13)$$

with $A > 0$ due to our assumption on α. For the update of y, we have

$$\begin{aligned} f(x_{t+1},y_{t+1}) &\leq f(x_{t+1},y_t) \\ &\quad - \beta\langle \nabla_y f(x_{t+1},y_t), \nabla_y g(x_{t+1},y_t)\rangle \\ &\quad + \frac{\beta^2 G_2(x)}{2}\|\nabla_y g(x_{t+1},y_t)\|^2. \end{aligned} \quad (14)$$

Using the assumption on the inner product, this yields

$$f(x_{t+1},y_{t+1}) \leq f(x_{t+1},y_t) - B\|\nabla_y f(x_{t+1},y_t)\|^2. \quad (15)$$

Combining the two equations, we get

$$\begin{aligned} &f(x_{t+1},y_{t+1}) \\ &\leq f(x_t,y_t) - A\|\nabla_x f(x_t,y_t)\|^2 - B\|\nabla_y f(x_{t+1},y_t)\|^2 \\ &\leq f(x_t,y_t) - C\left(\|\nabla_x f(x_t,y_t)\|^2 + \|\nabla_y f(x_{t+1},y_t)\|^2\right). \end{aligned} \quad (16)$$

with $C = \max(A,B)$. We define $G_t = \|\nabla_x f(x_t,y_t)\|^2 + \|\nabla_y f(x_{t+1},y_t)\|^2$ and rewrite this as

$$G_t \leq \frac{f(x_t,y_t) - f(x_{t+1},y_{t+1})}{C} \quad (17)$$

Summing this equation for $t = 0,\ldots,T$, we get

$$\sum_{t=0}^{T} G_t \leq \frac{f(x_0,y_0) - f(x_{T+1},y_{T+1})}{C}. \quad (18)$$

Since f is lower-bounded, this implies $G_t \to 0$, which in turn implies convergence to a first-order stationary point of f. □

Proof of Proposition 1. The gradient of f with respect to x is

$$\nabla_x f(x,y) = M(y)\nabla L_R(x) + (1-M(y))\nabla L_F(x) \quad (19)$$

Since $M(y)$ is bounded, $\nabla_x f$ is Lipschitz continuous in x given that L_R and L_F are Lipschitz smooth.

For the assumptions on the y-gradients, we fix x and treat the two cases in the definition of g separately. We only consider the case $L_R(x) < L_F(x)$ here, the reverse case is

completely analogous. Define $L(x) = L_F(x) - L_R(x) > 0$. The gradient of f with respect to y is

$$\nabla_y f(x,y) = -L(x)\nabla M(y) \tag{20}$$

and is Lipschitz continuous with constant $G_2(x) = |-L(x)|G = L(x)G$, where G is the Lipschitz constant of $M(y)$. We have

$$\nabla_y g(x,y) = -\left(L(x) + \frac{\lambda}{M(y)+\varepsilon}\right)\nabla M(y). \tag{21}$$

The inner product of the two gradients reads

$$\begin{aligned}&\langle \nabla_y f(x,y), \nabla_y g(x,y)\rangle \\ &= L(x)\left(L(x) + \frac{\lambda}{M(y)+\varepsilon}\right)\|\nabla M(y)\|^2,\end{aligned} \tag{22}$$

and for the gradient norms we get

$$\|\nabla_y f(x,y)\|^2 = L(x)^2\|\nabla M(y)\|^2, \tag{23}$$

as well as

$$\|\nabla_y g(x,y)\|^2 = \left(L(x) + \frac{\lambda}{M(y)+\varepsilon}\right)^2\|\nabla M(y)\|^2. \tag{24}$$

Plugging everything into the inner product assumption of Theorem 1 and simplifying yields

$$\begin{aligned}\beta\left(L(x) + \frac{\lambda}{M(y)+\varepsilon}\right) \geq \frac{G\beta^2}{2}\left(L(x) + \frac{\lambda}{M(y)+\varepsilon}\right)^2 \\ + BL(x)\end{aligned} \tag{25}$$

Since M, L_R and L_F are bounded, one easily finds a choice for β and B that satisfies this condition. □

A.2. The camera warping function w_c and static flow transformer ν

The network C predicts camera motion that consist of camera rotations $sin\alpha, sin\beta, sin\gamma$, and translations t_x, t_y, t_z. Thus $e = (sin\alpha, sin\beta, sin\gamma, t_x, t_y, t_z)$. Given camera motion and depth d, we transform the image coordinates (x, y) into world coordinates (X, Y, Z).

$$X = \tfrac{d}{f}(x - c_x) \tag{26}$$

$$Y = \tfrac{d}{f}(y - c_y) \tag{27}$$

$$Z = d \tag{28}$$

where (c_x, c_y, f) constitute the camera intrinsics. We now transform the world coordinates given the camera rotation and translation.

$$\mathbf{X}' = R_x R_y R_z \mathbf{X} + t$$

where $(R_x R_y R_z, t) \in SE3$ denote 3D rotation and translation, and $\mathbf{X} = [X, Y, Z]^T$. Hence, in image coordinates

$$x' = \frac{f}{Z} + c_x \tag{29}$$

$$y' = \frac{f}{Z} + c_y \tag{30}$$

We can now apply the warping as,

$$w_c\big(I(x,y), e, d\big) = I(x', y'). \tag{31}$$

The static flow transformer is defined as,

$$\nu(e,d) = (x' - x, y' - y) \tag{32}$$

A.3. The flow warping function, w_f

The flow warping function w_f is given by

$$w_f\big(I(x,y), u_x, u_y\big) = I(x + u_x, y + u_y) \tag{33}$$

where, (u_x, u_y) is the optical flow, and (x, y) is the spatial coordinate system.

A.4. Network Architectures

We briefly describe the network architectures below. For details, please refer to Figure 2.

Depth Network D. Our depth network is similar to DispNetS [7] and outputs depths at 6 different scales. Each convolution and upconvolution is followed by a ReLU except the prediction layers. The prediction layer at each scale has a non-linearity given by $1/(\alpha\ \text{sigmoid}(x) + \beta)$. The architecture of DispResNet is obtained by replacing convolutional blocks in DispNet by residual blocks [4].

Camera Motion Network C. The camera motion network consists of 8 convolutional layers, each of stride 2 followed by a ReLU activation. This is followed by a convolutional layer of stride 1, whose feature maps are averaged together to get the camera motion.

Flow Network F. We use FlowNetC architecture [2] with 6 output scales of flow and is shown in Figure 2. All convolutional and upconvolutional layers are followed by a ReLU except prediction layers. The prediction layers have no activations. For PWC Net, we use the network architecture from Janai et al. [5].

Mask Network M. The mask network has a U-Net [2] architecture. The encoder is similar to the camera motion with 6 convolutional layers. The decoder has 6 upconvolutional layers. Each of these layers have ReLU activations. The prediction layers use a sigmoid.

A.5. Qualitative Results

The qualitative results of the predictions are shown in Figure 3. We would like to point out that, our method is able to segment the moving car, and not the parked cars on the

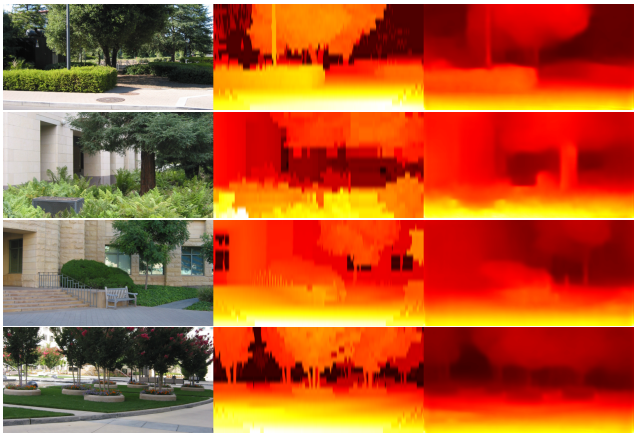

Figure 1: Qualitative results on Make3D test set.

roads. In addition, it segments other moving objects, such as the bicyclist.

We compare the qualitative results for single image depth prediction in Figure 4. We also contrast our results with basic models that were trained independently without a joint loss in Figure 5. We observe that our model produces better results, capturing moving objects such as cars and bikes, as well as surface edges of trees, pavements and buildings.

We compare the qualitative results for optical flow estimation in Figure 6. We show that our method performs better than UnFlow [8], Geonet [11] and DF-Net [13] . Our flow estimations are better at the boundaries of the cars and pavements. In contrast, competing methods produce blurry flow fields.

A.6. Additional Experiments

Depth evaluation on Make3D dataset. We also test on the Make3D dataset [10] without training on it. We use our our model that is trained only on Cityscapes and KITTI. Our method outperforms previous work [12, 13, 3] as shown in Table 3. We show qualitative results in Fig. 1.

Pose evaluation on Sintel. We test on Sintel's alley sequence [1] without training on it and compare it with Zhou et al. [12]. For this comparison, Zhou et al.'s model is taken from Pinard's implementation. We show quantitative evaluation using relative errors on pose in Table 4.

Training using a shared encoder. We train the camera motion network, C and motion segmentation network, M using a common shared encoder but different decoders. Intuitively, it seems that camera motion network can benefit from knowing static regions in a scene, which are learned by the motion segmentation network. However, we observe a

Zhou [12]	DF-Net [13]	Godard [3]	CC (ours)
0.383	0.331	0.361	**0.320**

Table 3: Absolute Relative errors on Make3D test set.

	alley 1	alley 2
Zhou et al. [12]	0.002 ± 0.001	0.027 ± 0.019
CC (ours)	0.002 ± 0.001	0.022 ± 0.015

Table 4: Relative errors on Sintel alley sequences.

	Sequence 09	Sequence 10
Shared Encoder	0.017 ± 0.009	0.015 ± 0.009
Uncoupled Networks	$\mathbf{0.012 \pm 0.007}$	$\mathbf{0.012 \pm 0.008}$

Table 5: Absolute Trajectory errors on KITTI Odometry.

Method	Depth	Pose	Flow	Mask
Geonet [11]	$15ms$	$4ms$	$45ms$	-
CC (ours)	$13ms$	$2ms$	$34ms$	$3ms$

Table 6: Average runtime on TitanX GPU with images of size 128×418.

performance degradation on camera motion estimates (Table 5). The degradation of results using a shared encoder are because feature encodings for one network might not be optimal for other networks. Our observation is consistent with Godard et al. [3] (Supp. Mat. Table 4), where sharing an encoder for depth and camera motion estimation improves depth but the perfomance on camera motion estimates are not as good.

A.7. Timing Analysis

We analyze inference time of our network and compare it with Geonet [11] in Table 6. We observe that our networks have a faster run time using the same sized 128 $\times$416 images on a single TitanX GPU. This is because our networks are simpler and smaller than ones used by Geonet.

For training, we measure the time taken for each iteration consisting of forward and backward pass using a batch size of 4. Training depth and camera motion networks (D, C) takes $0.96s$ per iteration. Traing the mask network, M takes $0.48s$ per iteration, and the flow network F takes $1.32s$ per iteration. All iterations have a batch size of 4. In total, it takes about 7 days for all the networks to train starting with random initialization on a single 16GB Tesla V100.

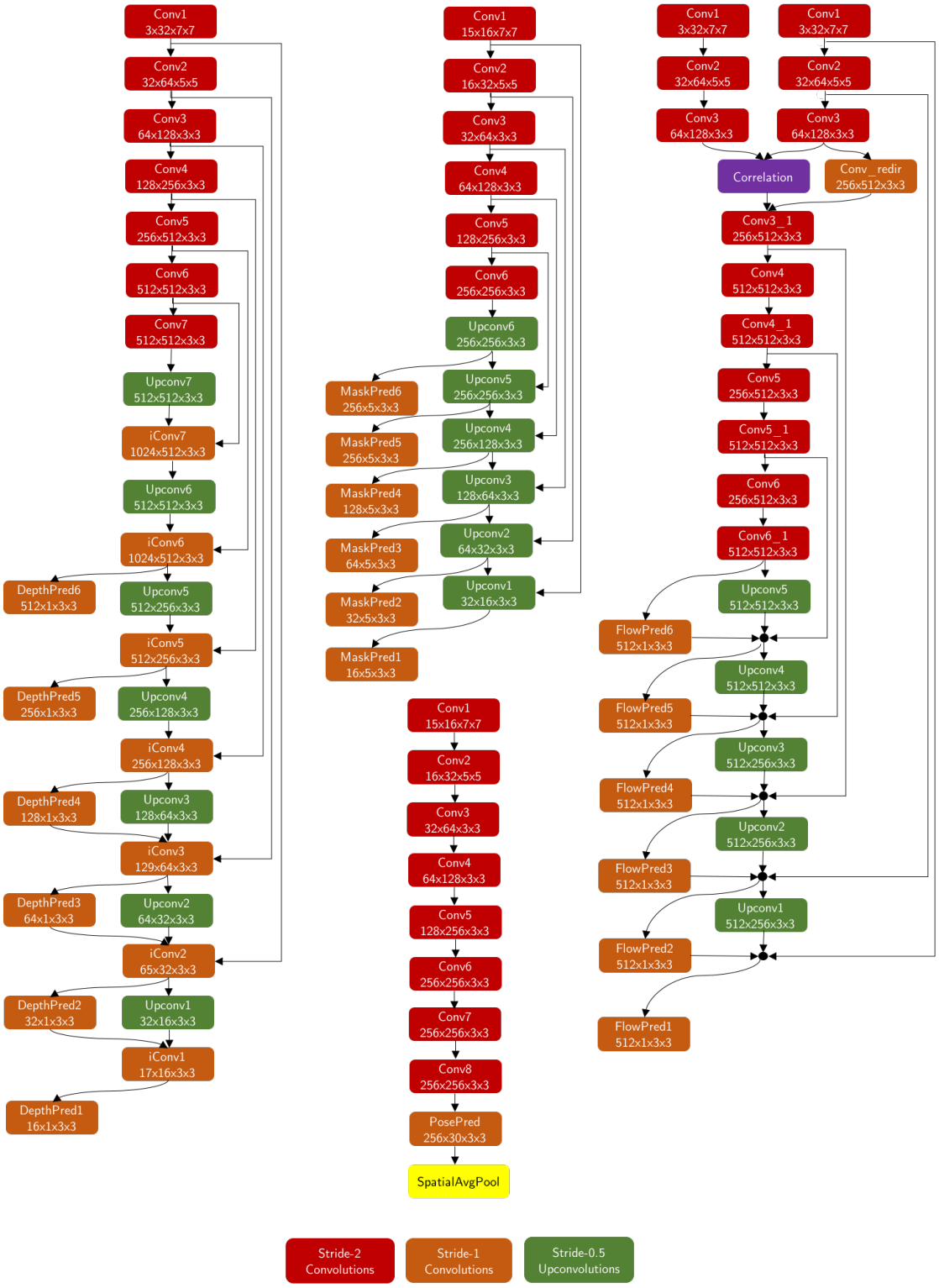

Figure 2: Architecture of the DispNet (left), MaskNet (center-top), FlowNetC (right) and Camera Motion Network (center-bottom). Convolutional layers are red (stride 2) and orange (stride 1) and upconvolution layers are green (stride 2). Other colors refer to special layers. Each layer is followed by ReLU, except prediction layers. In each block, the numbers indicate the number of channels of the input feature map, the number of channels of the output feature map, and the filter size.

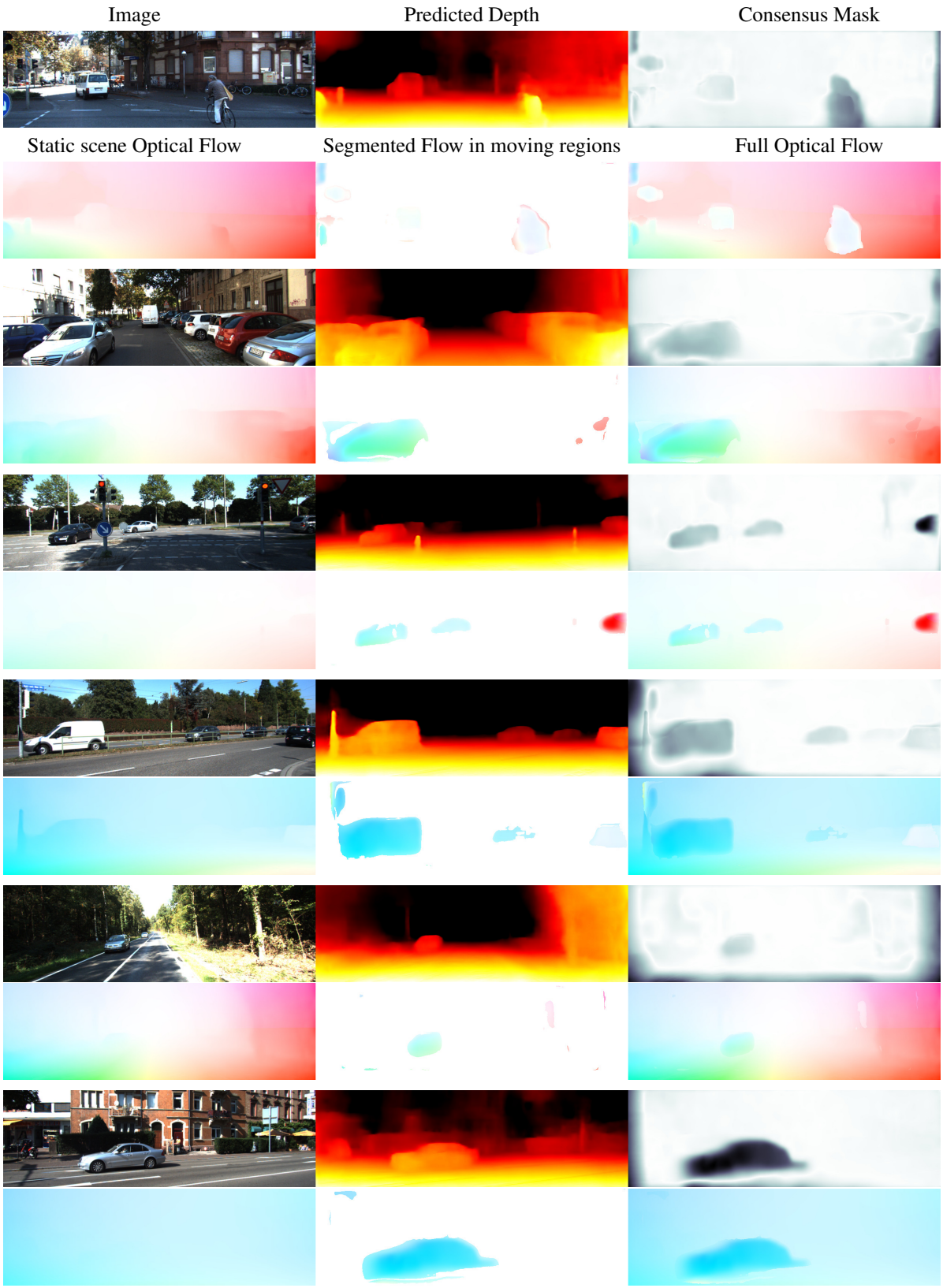

Figure 3: Network Predictions. Top row: we show image, predicted depth, consensus masks. Bottom row: we show static scene optical flow, segmented flow in the moving regions and full optical flow.

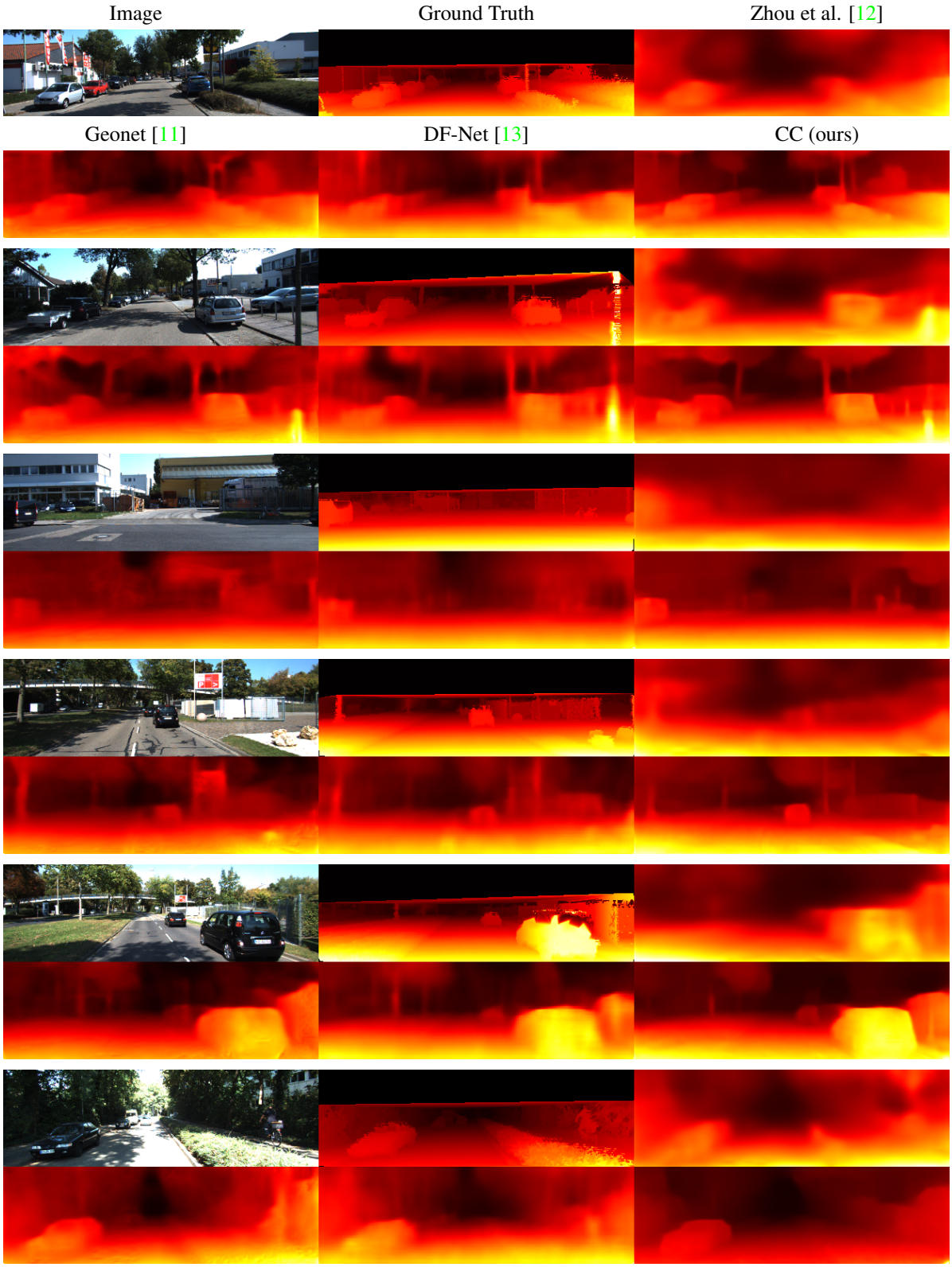

Figure 4: Qualitative results on single view depth prediction. Top row: we show image, interpolated ground truth depths, Zhou et al. [12] results. Bottom row: we show results from Geonet [11], DF-Net [13] and CC (ours) results.

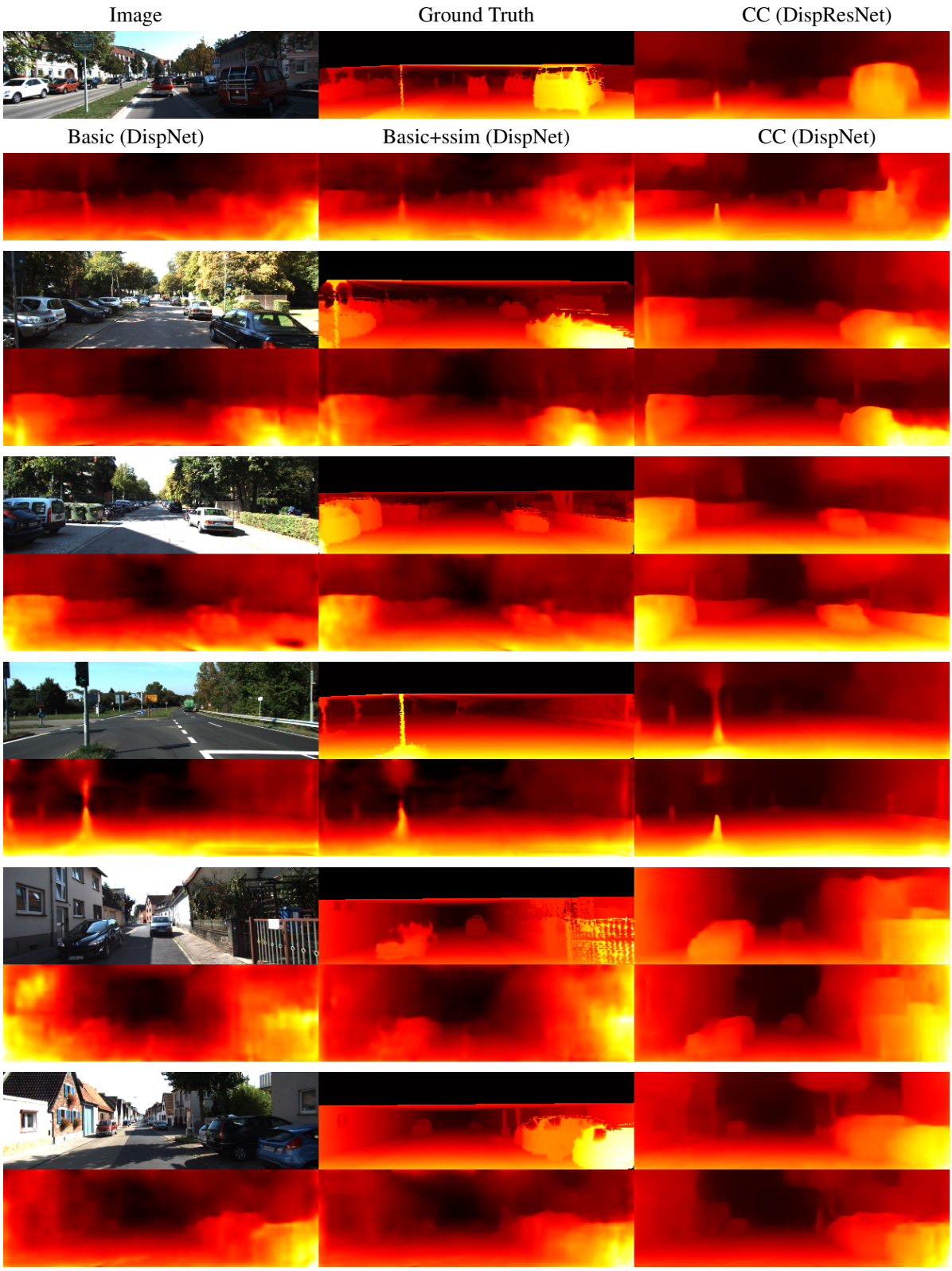

Figure 5: Ablation studies on single view depth prediction. Top row: we show image, interpolated ground truth depths, CC using DispResNet architecture. Bottom row: we show results using Basic, Basic+ssim and CC models using DispNet architecture.

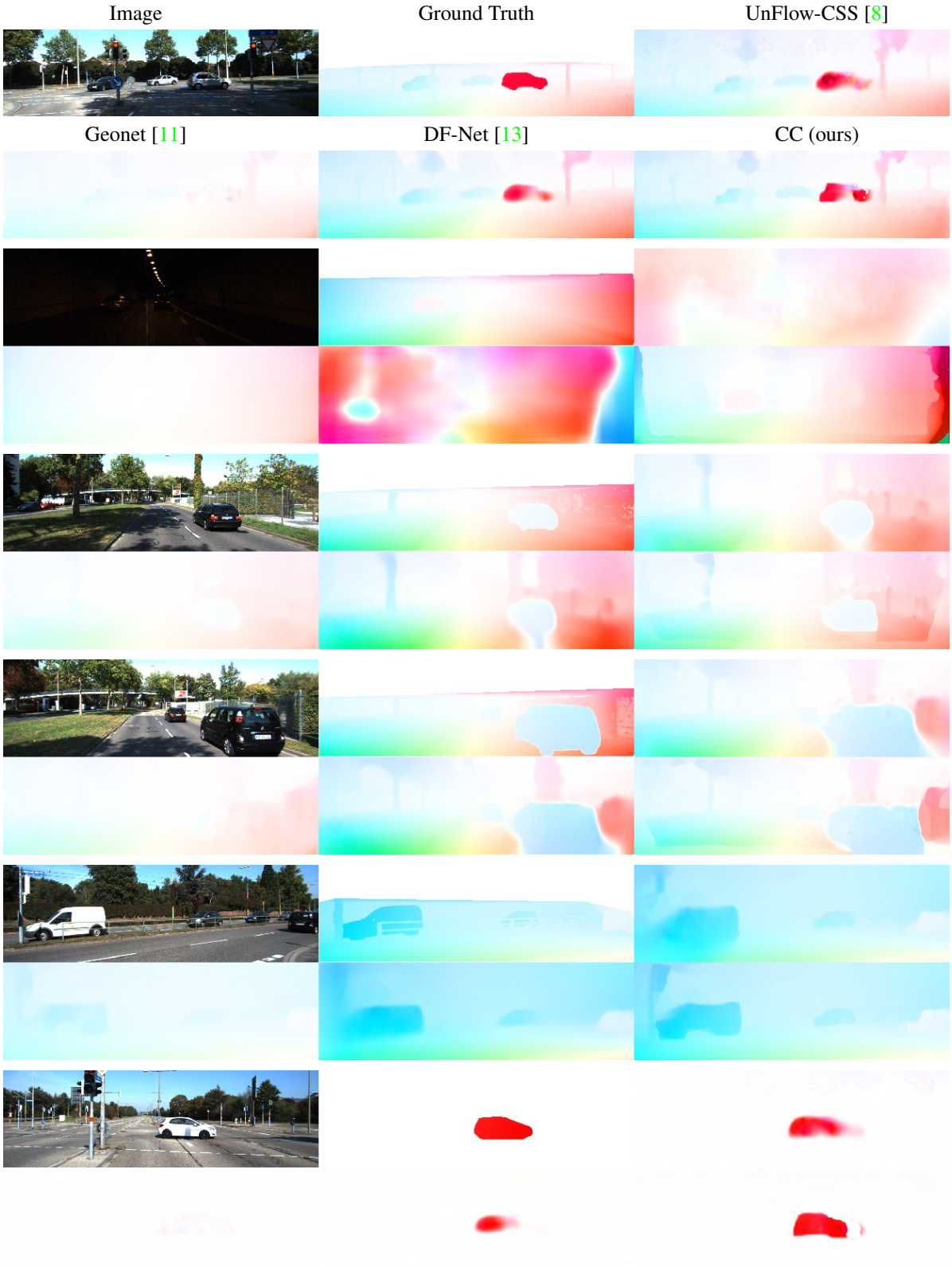

Figure 6: Qualitative results on Optical Flow estimation. Top row: we show image 1, ground truth flow, and predictions from UnFlow-CSS[8]. Bottom row: we show predictions from Geonet [11], DF-Net [13] and CC (ours) model.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625, 2012. 5

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 4

[3] C. Godard, O. Mac Aodha, and G. Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018. 5

[4] 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*, pages 770–778, 2016. 4

[5] J. Janai, F. Güney, A. Ranjan, M. Black, and A. Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 690–706, 2018. 4

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[7] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 4

[8] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. *arXiv preprint arXiv:1711.07837*, 2017. 5, 10

[9] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011. 1

[10] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006. 5

[11] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018. 5, 8, 10

[12] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017. 5, 8

[13] Y. Zou, Z. Luo, and J.-B. Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *European Conference on Computer Vision*, pages 38–55. Springer, 2018. 5, 8, 10