Open access • Proceedings Article • DOI:10.1109/CVPR.2012.6247873

Layered segmentation and optical flow estimation over time — Source link

Deqing Sun, Erik B. Sudderth, Michael J. Black

Institutions: Brown University

Published on: 16 Jun 2012 - Computer Vision and Pattern Recognition

Topics: Image segmentation, Motion estimation, Optical flow, Continuous optimization and Gradient descent

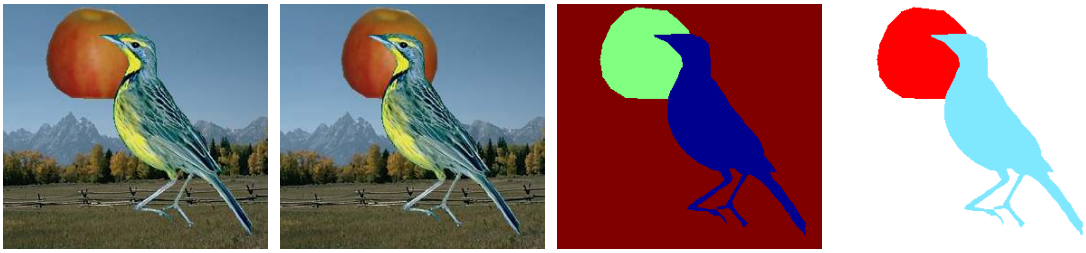
Layered Segmentation and Optical Flow Estimation Over Time: Supplemental Material

Deqing Sun[1] Erik B. Sudderth[1] Michael J. Black[1,2]
[1]Department of Computer Science, Brown University, Providence, RI 02912, USA
[2]Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany

Section 1 provides additional details for each of the "cooperative" moves used in optimizing the discrete layered model. Section 2 provides a high-level description of the algorithms for determining the depth ordering and the number of layers. Section 3 provides the full set of images illustrating the experimental results. The results include the screen shots of the Middlebury AAE and EPE evaluation tables at the time of writing (April 2012), as well as all experimental results on the Middlebury flow dataset and the MIT layer segmentation dataset.

1. "Cooperative" Moves for the Discrete Model

We will use a toy example to explain the effect achieved by each move. Figure 1 shows the desired layer segmentation and flow field for the input "bird apple" sequence. During optimization we will see that there are several (fairly bad) local optima and we will need to make large changes to the solution to get out of these optima. Note that the binary selection variable $\mathbf{b}$ controls different variables for each move. The potential functions for each move are also defined differently though the functions may share the same name.



(a) first image (b) second image (c) segmentation (d) flow field

Figure 1. Input "bird apple" frames, the ground truth segmentation, and flow field.

Simultaneous segmentation and flow move. Sometimes a region may be assigned to the wrong layer but with the correct motion, as shown in Figure 2. Changing the segmentation or the flow field alone will not move the solution from the local optimum. We need to simultaneously change the segmentation and the motion. At each step, this move allows a pixel to become visible at a particular layer $\hat{k}$. The newly visible pixel at layer $\hat{k}$ takes the motion of the corresponding pixel at the previously visible layer k', while the latter takes the motion of its affine flow field. Note that the binary decision variable b_t^p may influence all the support functions for the first $\hat{k}$ layers. When the binary decision variable is 0, the configuration is unchanged. When it is 1, $u^p_{t\hat{k}}(1) = u^{p,\text{old}}_{tk'}, u^p_{tk'}(1) = u^p_{\theta_{tk'}}, g^p_{tk}(1) = 0, k < \hat{k}$, and $g^p_{t\hat{k}}(1) = 1$. Using the binary variable

(a) segmentation (b) flow field

Figure 2. Local minimum of the objective requiring the simultaneous segmentation and flow move. A large region of pixels in the center has been incorrectly assigned to the bird layer though their motion is correct. Changing either the segmentation or the flow field alone will not resolve this and get us out of the local minimum. We need to make these pixels visible in the apple layer and still retain their motion using the simultaneous segmentation and flow move (see text).

representation, we can perform the following simplification

$$\begin{aligned} E(\mathbf{b}) = &\sum_{t=1}^{T-1} \Big\{ E_{\text{data}}(\mathbf{u}_t(\mathbf{b}_t), \mathbf{v}_t(\mathbf{b}_t), \mathbf{g}_t(\mathbf{b}_t), \mathbf{g}_{t+1}(\mathbf{b}_{t+1})) + \sum_{k=1}^{K} \lambda_a (E_{\text{space}}^{\text{flow}}(\mathbf{u}_{tk}(\mathbf{b}_t), \theta_{tk}) + E_{\text{space}}^{\text{flow}}(\mathbf{v}_{tk}(\mathbf{b}_t), \theta_{tk})) \\ &+ \sum_{k=1}^{K-1} \lambda_b E_{\text{space}}^{\text{sup}}(\mathbf{g}_{tk}(\mathbf{b}_t)) + \lambda_c E_{\text{time}}(\mathbf{g}_{tk}(\mathbf{b}_t), \mathbf{g}_{t+1,k}(\mathbf{b}_{t+1}), \mathbf{u}_{tk}(\mathbf{b}_t), \mathbf{v}_{tk}(\mathbf{b}_t)) \Big\} + \sum_{k=1}^{K-1} \lambda_b E_{\text{space}}^{\text{sup}}(\mathbf{g}_{Tk}(\mathbf{b}_T)) \quad (1) \\ = &\sum_{t=1}^{T-1} \sum_{p} \left(\sum_{q \in \mathcal{N}_{p,\text{time}}^0} \phi_{\text{time}}^0(b_t^p, b_{t+1}^q) + \sum_{q \in \mathcal{N}_{p,\text{time}}^1} \phi_{\text{time}}^1(b_t^p, b_{t+1}^q) \right) + \sum_{t=1}^{T} \sum_{p} \left[\sum_{q \in \mathcal{N}_{p,\text{space}}} \phi_{\text{space}}(b_t^p, b_t^q) + \phi_{\text{affine}}(b_t^p) \right], \end{aligned}$$

where $\mathbf{u}_t(\mathbf{b}_t)$ and $\mathbf{v}_t(\mathbf{b}_t)$ are deterministic functions of the candidate flow fields and the binary selection variable $\mathbf{b}_t$, and $\mathbf{g}_t(\mathbf{b}_t)$ is a deterministic function of the previous support function $\mathbf{g}_t^{\text{old}}$ and the binary selection variable $\mathbf{b}_t$.

A pixel p at each layer has two flow vector candidates to select and may interact with two different pixels at the next frame. We define different potential functions for both temporal neighborhood structures. For example, when $b_t^p = 0$, the temporal neighbors are

$$\mathcal{N}_{p,\text{time}}^0 = \{(i + [u_{tk}^p(0)], j + [v_{tk}^p(0)]), 1 \le k \le K\}, \quad (2)$$

and the corresponding potential function will only "fire" when $b_t^p = 0$, *i.e.*

$$\phi_{\text{time}}^0(b_t^p, b_{t+1}^q) = \Big[\Big(\rho_d(\mathbf{I}_t^p - \mathbf{I}_{t+1}^{q'}) - \lambda_d \Big) s_{tk}^p(b_t^p) s_{t+1,k}^q(b_{t+1}^q) + \lambda_c (1 - \delta(g_{tk}^p(b_t^p), g_{t+1,k}^q(b_{t+1}^q)))(1 - \delta(k, K)) \Big] (1 - b_t^p), \quad (3)$$

which incorporates both the data term and the temporal consistency term of the first $K-1$ support functions. We evaluate the warped image $I_{t+1}^{q'}$ at subpixel positions and the visibility mask $s_{t+1,k}^q$ and the warped support function $g_{t+1,k}^q$ at integer positions. Similarly

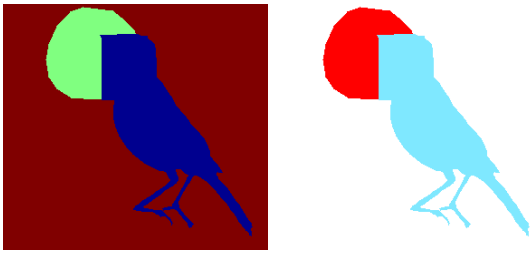
$$\mathcal{N}_{p,\text{time}}^1 = \{(i + [u_{tk}^p(1)], j + [v_{tk}^p(1)]), 1 \le k \le K\}, \quad (4)$$

and $\phi_{\text{time}}^1(b_t^p, b_{t+1}^q)$ only "fires" when $b_t^p = 1$

$$\phi_{\text{time}}^1(b_t^p, b_{t+1}^q) = \Big[\Big(\rho_d(\mathbf{I}_t^p - \mathbf{I}_{t+1}^{q'}) - \lambda_d \Big) s_{tk}^p(b_t^p) s_{t+1,k}^q(b_{t+1}^q) + \lambda_c (1 - \delta(g_{tk}^p(b_t^p), g_{t+1,k}^q(b_{t+1}^q)))(1 - \delta(k, K)) \Big] b_t^p. \quad (5)$$

For the spatial pairwise term, the set $\mathcal{N}_{p,\text{space}}$ contains all the four nearest neighbors of the pixel p and is fixed throughout the document. The binary selection variable changes the states of several binary support functions and flow fields. The effects sum together as

$$\phi_{\text{space}}(b_t^p, b_t^q) = \sum_{k=1}^{K-1} \lambda_b w_q^p \big(1 - \delta(g_{tk}^p(b_t^p), g_{tk}^q(b_t^q))\big) + \sum_{k=1}^{K} \lambda_a \rho_{\text{mrf}}\big(u_{tk}^p(b_t^p) - u_{tk}^q(b_t^q)\big). \quad (6)$$



(a) segmentation (b) flow field

Figure 3. Local minimum that can be solved with the visibility move. A large number of pixels in the center of the image are assigned incorrectly to the bird layer, while the apple layer has the correct motion field for these pixels. Changing these pixels to be visible at the apple layer will fix this particular solution and get us out of the local minimum.

The unary term enforces the selected motion vector to be similar to the affine motion field of the newly visible layer

$$\phi_{\text{affine}}(b_t^p) = \sum_{k=1}^{K} \lambda_a \lambda_{\text{aff}} \rho_{\text{aff}}(u_{tk}^p(b_t^p) - u_{\theta_{tk}}^p). \tag{7}$$

We can solve this binary problem using QPBO and set the states of all these involved variables according to the binary selection variables.

Visibility move. Sometimes the segmentation is wrong, while the correct layer has the right motion field. Hence we can make big changes to the layer segmentation to correct the errors in the motion field, as shown in Figure 3. Given the current flow estimate, we decide whether to make a pixel p visible for some layer $\hat{k}$ by modifying the previous layer support $\mathbf{g}^{\text{old}}$. Because the visibility state of a pixel is jointly determined by several support functions, the binary selection variable controls the states of all the support functions involved. When $b_t^p = 0$, all the support functions retain their previous value at p, *i.e.* $g_{tk}^p(0) = g_{tk}^{p,\text{old}}$. When $b_t^p = 1$, we need to adjust the support functions of the first $\hat{k}$ layers so that layer $\hat{k}$ is visible at p, *i.e.* $g_{tk}^p(1) = 0$ if $k < \hat{k}$, and $g_{tk}^p(1) = 1$ if $k = \hat{k}$. When $\hat{k}$ is the last layer, all the support functions of the first $K-1$ layers are set to be 0 at p. The energy function for the binary variable is

$$\begin{aligned} E(\mathbf{b}) &= \sum_{t=1}^{T-1} \Big\{ E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t(\mathbf{b}_t), \mathbf{g}_{t+1}(\mathbf{b}_{t+1})) + \lambda_c E_{\text{time}}(\mathbf{g}_{tk}(\mathbf{b}_t), \mathbf{g}_{t+1,k}(\mathbf{b}_{t+1}), \mathbf{u}_{tk}, \mathbf{v}_{tk}) \Big\} + \sum_{t=1}^{T} \sum_{k=1}^{K-1} \lambda_b E_{\text{space}}^{\text{sup}}(\mathbf{g}_{tk}(\mathbf{b}_t)) \\ &= \sum_{t=1}^{T-1} \sum_{p} \sum_{q \in \mathcal{N}_{p,\text{time}}} \phi_{\text{time}}(b_t^p, b_{t+1}^q) + \sum_{t=1}^{T} \sum_{p} \sum_{q \in \mathcal{N}_{p,\text{space}}} \phi_{\text{space}}(b_t^p, b_t^q), \end{aligned} \tag{8}$$

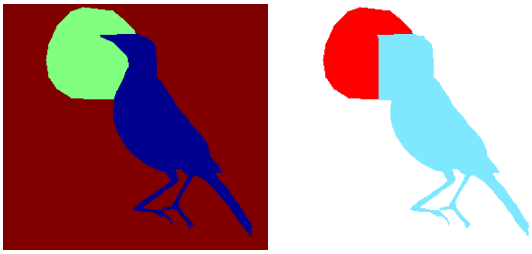
in which $\mathcal{N}_{p,\text{time}} = \{(i + [u_{tk}^p], j + [v_{tk}^p]), 1 \le k \le K\}$ and the time term incorporates both the data term and the temporal consistency of the first $K-1$ support functions

$$\phi_{\text{time}}(b_t^p, b_{t+1}^q) = \begin{cases} \Big(\rho_d(\mathbf{I}_t^p - \mathbf{I}_{t+1}^{q'}) - \lambda_d\Big) s_{tk}^p(b_t^p) s_{t+1,k}^q(b_{t+1}^q) + \lambda_c(1 - \delta(g_{tk}^p(b_t^p), g_{t+1,k}^q(b_{t+1}^q))), & k < K, \\ \Big(\rho_d(\mathbf{I}_t^p - \mathbf{I}_{t+1}^{q'}) - \lambda_d\Big) s_{tk}^p(b_t^p) s_{t+1,k}^q(b_{t+1}^q), & k = K, \end{cases} \tag{9}$$

where the visibility mask s depends on the support function which in turn depends on the binary variable b, and the corresponding pixel q depends on the flow vector of the kth layer. The visibility move may change several support functions and the potential function for the spatial term is

$$\phi_{\text{space}}(b_t^p, b_t^q) = \sum_{k=1}^{K-1} \lambda_b w_q^p \Big(1 - \delta(g_{tk}^p(b_t^p), g_{tk}^q(b_t^q))\Big). \tag{10}$$

Support function move. Given the current flow estimate, we decide whether to make the support function of a pixel p at the selected layer $\hat{k}$ equal to 1 or retain its previous value. The energy function and the potential functions are the same as Eqs. (8)- (10) but the binary variable selects the solutions in a different way. When $b_t^p = 0$, $g_{t\hat{k}}^p(0) = g_{t\hat{k}}^{p,\text{old}}$. When $b_t^p = 1$, $g_{t\hat{k}}^p(1) = 1$.



(a) segmentation (b) flow field

Figure 4. Local minimum that can be solved using the occlusion-aware fusion flow move. A large number of pixels in the center of the image have the wrong motion vectors. Making large changes to the flow field can reduce the errors. Note that we use the segmentation information to reason about occlusions.

Occlusion-aware fusion flow move. Here we deal with the case in which the segmentation is correct, but the motion field has a large region of errors, particularly in occlusion regions, as shown in Figure 4. We must make large changes to the flow field to reduce the motion errors. Given the current estimate of the support functions for each layer, we want to select the motion of each pixel between its current flow estimate and its affine mean flow field, $u^p_{tk}(0) = u^{p,\text{old}}_{tk}$ and $u^p_{tk}(1) = u^p_{\theta_{tk}}$. Note that for occluded pixels, their data likelihood term does not provide useful information to estimate the motion. Instead we can predict the motion of the occluded pixels using the affine motion field fitted to the visible pixels of the same object (layer). For a particular layer k of frame t ($t < T$), the energy function to minimize is

$$\begin{aligned} E(\mathbf{b}_{tk}) =& E_{\text{data}}(\mathbf{u}_t(\mathbf{b}_{tk}), \mathbf{v}_t(\mathbf{b}_{tk}), \mathbf{g}_t, \mathbf{g}_{t+1}) + \sum_{k=1}^{K} \lambda_a (E^{\text{flow}}_{\text{space}}(\mathbf{u}_{tk}(\mathbf{b}_{tk}), \theta_{tk}) + E^{\text{flow}}_{\text{space}}(\mathbf{v}_{tk}(\mathbf{b}_{tk}), \theta_{tk})) \\ &+ \lambda_c \sum_{k=1}^{K-1} E_{\text{time}}(\mathbf{g}_{tk}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}(\mathbf{b}_{tk}), \mathbf{v}_{tk}(\mathbf{b}_{tk})) \\ =& \sum_p \phi_{\text{unary}}(b^p_{tk}) + \sum_p \sum_{q \in \mathcal{N}_{p,\text{space}}} \phi_{\text{space}}(b^p_{tk}, b^q_{tk}), \end{aligned} \tag{11}$$

where the unary term incorporates the data term, the temporal consistency of the support function, and the deviation from the affine motion field

$$\phi_{\text{unary}}(b^p_{tk}) = \Big(\rho_d(\mathbf{I}^p_t - \mathbf{I}^{q'}_{t+1}) - \lambda_d\Big) s^p_{tk} + \lambda_c(1 - \delta(g^p_{tk}, g^q_{t+1,k}))(1 - \delta(k, K)) + \lambda_a \lambda_{\text{aff}} \rho_{\text{aff}}(u^p_{tk}(b^p_t) - u^p_{\theta_{tk}}), \tag{12}$$

in which the corresponding pixel at the next frame depends on the flow field selection at the current frame, $q' = \big(i + u^p_{tk}(b^p_{tk}), j + v^p_{tk}(b^p_{tk})\big)$ and $q = \big(i + [u^p_{tk}(b^p_{tk})], j + [v^p_{tk}(b^p_{tk})]\big)$. This energy function differs from the FusionFlow method [3] in that the segmentation information directly modulates the data likelihood term in Eq. (12) and enables occlusion reasoning.

Continuous flow refinement. Given the current estimate of the support functions for each layer, we refine the flow field for each layer by minimizing $E(\mathbf{u}_{tk}, \mathbf{v}_{tk}) =$

$$E_{\text{data}}(\mathbf{u}_{tk}, \mathbf{v}_{tk}, \mathbf{g}_t, \mathbf{g}_{t+1}) + \lambda_a (E^{\text{flow}}_{\text{space}}(\mathbf{u}_{tk}, \theta_{tk}) + E^{\text{flow}}_{\text{space}}(\mathbf{v}_{tk}, \theta_{tk})) + \lambda_c E_{\text{time}}(\mathbf{g}_{tk}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}, \mathbf{v}_{tk})(1 - \delta(k, K)). \tag{13}$$

Compared with standard optical flow formulation, this energy function contains the segmentation information necessary to reason about occlusions and an additional temporal consistency term for the support functions. We can still optimize the energy function using standard warping-based incremental estimation methods [4] and the refinement step adaptively changes the flow field for the discrete optimization.

2. Pseudo Code for the Discrete Optimizer

The high-level algorithm for the initialization and the discrete optimization is

```
Perform flow estimation using "Classic+NL"
Cluster initial flow into different affine motion groups
```

```
Perform Sequence A for forward and backward depth ordering
Pick the solution with lower energy
Perform Sequence B to decide depth ordering
Perform Sequence A to refine segmentation and flow
```

Sequence A tries to merge layers with similar motion together and includes

```
Perform visibility move for each layer
Remove redundant layers
Perform joint segmentation and flow move
Remove redundant layers
Perform support function move
Remove redundant layers
Re-estimate affine flow field for each layer
Perform flow selection move
Perform continuous flow refinement
```

During the optimization, some layers may have no visible pixels and are redundant in the energy minimization. Removing these layers will always reduce the energy of the solution. The new solution can explain the image data equally well but does not pay any cost for the spatial and temporal penalty terms for the redundant layers. Hence we remove these redundant layers and change the number of layers accordingly.

Sequence B determines the local depth ordering between neighboring layers and includes

```
For iter = 1:maxIters
  Select candidate pairs to compare by comparing the occlusion area
  current_solution = input_solution
  local_minimum = true
  For each selected pair of selected neighboring layers
    Swap depth ordering
    Perform visibility move for the two selected layers
    Perform support function move for the two selected layers
    If energy(new solution) < energy(current solution)
      current_solution = new_solution
      local_minimum = false
    End if
  End for
  Remove redundant layers
  If local_minimum == true
    break
  End if
End for
```

maxIters is set to be 10 in the current implementation and the algorithm usually stops after $4 - 6$ iterations.

The pseudo code for selecting the candidate neighboring layer pairs is

```
Input: max_pairs (max number of layer pairs to compare)
Output: pairs of neighboring layers to compare
Compute (num_pairs) neighboring pairs using current segmentation
If num_pairs <= max_pairs
   Output all the neighboring pairs and exit
Else
   Compute occlusion regions between each neighboring pairs
   Order the pairs by the occlusion area in descending order
   Output the top max_pairs neighboring layer pairs
End of if
```

3. Additional Experimental Results

Figure 5 provides a screen shot of the top-performing methods on the Middlebury evaluation website at the time of writing (April 2012). Figure 6 shows the color keys for the ordering of layers and the flow field. Figures 7-10 show the first frame, estimated layer segmentation, and the estimated flow fields on the Middlebury optical flow training and test sets. Figure 11 shows the first frame, the segmentations by **HGVS** [2], **Layers++** [5], and **nLayers**, the flow field estimated by **nLayers**, and the human labeled ground truth.

Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
IROF++ [63]	7.8	0.08 3	0.23 5	0.07 7	0.21 14	0.68 13	0.17 16	0.28 10	0.63 8	0.19 16	0.15 7	0.73 9	0.09 10	0.60 3	0.89 4	0.42 5	0.43 9	1.08 9	0.31 9	0.10 4	0.12 5	0.12 3	0.47 4	0.98 6	0.68 9
Layers++ [38]	8.0	0.08 3	0.21 2	0.07 7	0.19 6	0.56 3	0.17 16	**0.20** 1	**0.40** 1	0.18 10	0.13 2	**0.58** 1	0.07 2	**0.48** 1	**0.70** 1	0.33 2	0.47 16	1.01 5	0.33 15	0.15 31	0.14 24	0.24 27	0.46 2	**0.88** 1	0.72 12
MDP-Flow2 [40]	8.3	0.09 13	0.23 5	0.07 7	0.16 2	**0.52** 1	0.13 3	0.22 2	0.46 3	0.17 6	0.17 19	0.93 27	0.09 10	0.65 7	0.98 9	0.43 6	**0.29** 1	0.91 3	0.26 3	0.11 8	0.13 13	0.17 10	0.51 14	1.11 16	0.72 12
nLayers [61]	8.5	**0.07** 1	**0.19** 1	**0.06** 1	0.22 17	0.59 4	0.19 26	0.25 6	0.54 5	0.20 24	0.15 7	0.84 17	0.08 5	0.53 2	0.78 2	0.34 3	0.44 11	**0.84** 1	0.30 7	0.13 18	0.13 13	0.20 17	0.47 4	0.97 5	0.67 7
Sparse-NonSparse [59]	11.8	0.08 3	0.23 5	0.07 7	0.22 17	0.73 19	0.18 19	0.28 10	0.64 9	0.19 16	0.14 4	0.71 6	0.08 5	0.67 11	0.99 11	0.48 10	0.49 19	1.06 7	0.32 12	0.14 24	**0.11** 1	0.28 39	0.49 8	0.98 6	0.73 16
COFM [64]	12.0	0.08 3	0.26 20	**0.06** 1	0.18 3	0.62 8	0.14 6	0.30 16	0.74 17	0.19 16	0.15 7	0.86 19	0.07 2	0.79 19	1.14 19	0.74 30	0.35 4	0.87 2	0.28 5	0.14 24	0.12 5	0.28 39	0.49 8	0.94 3	0.71 11
Efficient-NL [66]	12.0	0.08 3	0.22 4	**0.06** 1	0.23 27	0.73 19	0.18 19	0.32 20	0.75 18	0.18 10	0.14 4	0.72 8	0.08 5	0.60 3	0.88 3	0.43 6	0.57 26	1.11 12	0.35 21	0.14 24	0.13 13	0.25 30	0.48 6	0.90 2	0.63 5
TC-Flow [48]	12.5	**0.07** 1	0.21 2	**0.06** 1	**0.15** 1	0.59 4	**0.11** 1	0.31 18	0.78 20	**0.14** 1	0.16 14	0.86 19	0.08 5	0.75 16	1.11 17	0.54 15	0.42 8	1.40 22	0.25 2	0.11 8	0.12 5	0.29 43	0.62 21	1.35 22	0.93 33
LSM [41]	12.7	0.08 3	0.23 5	0.07 7	0.22 17	0.73 19	0.18 19	0.28 10	0.64 9	0.19 16	0.14 4	0.70 4	0.09 10	0.66 8	0.97 7	0.48 10	0.50 20	1.06 7	0.33 15	0.15 31	0.12 5	0.29 43	0.50 11	0.99 9	0.73 16
Ramp [68]	12.8	0.08 3	0.24 11	0.07 7	0.21 14	0.72 17	0.18 19	0.27 7	0.62 7	0.19 16	0.15 7	0.71 6	0.09 10	0.66 8	0.97 7	0.49 12	0.51 22	1.09 10	0.34 20	0.15 31	0.12 5	0.30 47	0.48 6	0.96 4	0.72 12
Classic+NL [31]	14.3	0.08 3	0.23 5	0.07 7	0.22 17	0.74 23	0.18 19	0.29 14	0.65 14	0.19 16	0.15 7	0.73 9	0.09 10	0.64 6	0.93 6	0.47 8	0.52 23	1.12 13	0.33 15	0.16 39	0.13 13	0.29 43	0.49 8	0.98 6	0.74 19
IROF-TV [56]	15.9	0.09 13	0.25 15	0.08 18	0.22 17	0.77 24	0.19 26	0.30 16	0.70 16	0.19 16	0.18 22	0.93 27	0.11 24	0.73 13	1.04 12	0.56 18	0.44 11	1.69 37	0.31 9	0.09 3	**0.11** 1	0.12 3	0.50 11	1.08 14	0.73 16

Average angle error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
nLayers [61]	5.7	**2.80** 1	**7.42** 1	**2.20** 1	2.71 12	7.24 2	2.55 27	2.61 2	6.24 2	2.45 25	2.30 3	12.7 4	1.16 3	**2.30** 1	**3.02** 1	**1.70** 1	2.62 2	**6.95** 1	2.09 2	2.29 14	3.46 10	1.89 15	1.38 2	3.06 3	1.29 2
Layers++ [38]	9.2	3.11 4	8.22 3	2.79 14	2.43 6	**7.02** 1	2.24 13	**2.43** 1	**5.77** 1	2.18 15	**2.13** 1	**9.71** 1	1.15 2	2.35 2	**3.02** 1	1.96 2	3.81 19	11.4 16	3.22 22	2.74 29	4.01 31	2.35 23	1.45 3	3.05 2	1.79 8
IROF++ [63]	10.1	3.17 8	8.69 6	2.61 7	2.79 13	9.61 13	2.33 17	3.43 9	8.86 11	2.38 20	2.87 12	14.8 12	1.52 14	2.74 5	3.57 5	2.19 6	3.20 7	9.70 9	2.71 9	1.96 7	3.45 9	1.22 5	1.80 10	4.06 11	2.50 17
MDP-Flow2 [40]	10.4	3.32 14	8.76 9	2.85 16	**2.18** 1	7.47 4	1.85 5	2.77 4	6.95 4	2.06 11	3.25 22	17.3 26	1.59 20	2.87 11	3.73 9	2.32 9	3.15 5	11.1 15	2.65 5	2.04 8	3.64 16	1.60 8	1.88 11	4.49 13	1.49 3
Sparse-NonSparse [59]	11.1	3.14 7	8.75 8	2.76 11	3.02 22	10.6 20	2.43 21	3.45 12	8.96 12	2.36 18	2.66 6	13.7 7	1.42 5	2.85 10	3.75 11	2.33 10	3.28 8	9.40 6	2.73 10	2.42 18	3.31 4	2.69 28	1.47 4	3.07 4	1.66 5
Efficient-NL [66]	11.2	3.01 3	8.29 4	2.30 3	3.12 25	10.3 18	2.40 18	3.83 19	9.97 18	2.08 12	2.76 10	14.4 10	1.45 8	2.64 4	3.51 4	2.07 5	3.06 3	8.23 2	2.49 4	2.53 22	3.73 17	2.46 25	1.91 12	3.32 6	2.40 16
LSM [41]	12.0	3.12 5	8.62 5	2.75 10	3.00 21	10.5 19	2.44 22	3.43 9	8.85 10	2.35 17	2.66 6	13.6 6	1.44 6	2.82 7	3.68 6	2.36 12	3.38 11	9.41 7	2.81 14	2.69 27	3.52 14	2.84 32	1.59 6	3.38 8	1.80 9
Ramp [68]	12.6	3.18 10	8.83 11	2.73 9	2.89 18	10.1 17	2.44 22	3.27 7	8.43 7	2.38 20	2.74 9	14.2 8	1.46 10	2.82 7	3.69 8	2.29 7	3.37 10	9.31 5	2.93 16	2.62 25	3.38 7	3.19 46	1.54 5	3.21 5	2.24 13
TC-Flow [48]	12.7	2.91 2	8.00 2	2.34 5	**2.18** 1	8.77 9	**1.52** 1	3.84 20	10.7 21	1.49 2	3.13 18	16.6 22	1.46 10	2.78 6	3.73 9	1.96 2	3.08 4	11.4 16	2.66 6	1.94 6	3.43 8	3.20 47	3.06 23	7.04 23	4.08 41
COFM [64]	14.0	3.17 8	9.90 24	2.46 6	2.41 4	8.34 8	1.92 6	3.77 17	10.5 19	2.54 27	2.71 8	14.9 14	1.19 4	3.08 15	3.92 15	3.25 36	3.83 20	10.9 12	3.15 19	2.20 12	3.35 5	2.91 39	1.62 8	**2.56** 1	2.09 10
Classic+NL [31]	14.3	3.20 12	8.72 7	2.81 15	3.02 22	10.6 20	2.44 22	3.46 13	8.84 9	2.38 20	2.78 11	14.3 9	1.46 10	2.83 9	3.68 6	2.31 8	3.40 12	9.09 3	2.76 13	2.87 34	3.82 21	2.86 35	1.67 9	3.53 9	2.26 15
SimpleFlow [52]	17.0	3.35 15	9.20 14	2.98 20	3.18 26	10.7 23	2.71 32	5.06 28	12.6 26	2.70 28	2.95 14	15.1 15	1.58 18	2.91 12	3.79 12	2.47 17	3.59 16	9.49 8	2.99 17	2.39 16	3.46 10	2.24 21	1.60 7	3.56 10	1.57 4

Figure 5. Screen shots of the Middlebury evaluation EPE/AAE tables. At the time of writing (April 2012), the proposed method, **nLayers**, is ranked fourth in EPE and first in AAE. Note that the EPE ranks for the top four methods are very close; while the AAE rank of **nLayers** is significantly higher than the other methods.

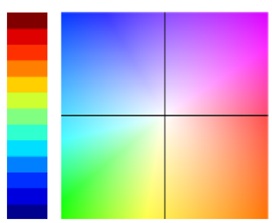


Figure 6. Left: the color key for the ordering of depth layers (blue is close and red is far); right: the color key for the flow fields [1].

References

[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, Mar. 2011. 6

[2] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. *CVPR*, pp. 2141–2148, 2010. 6, 11

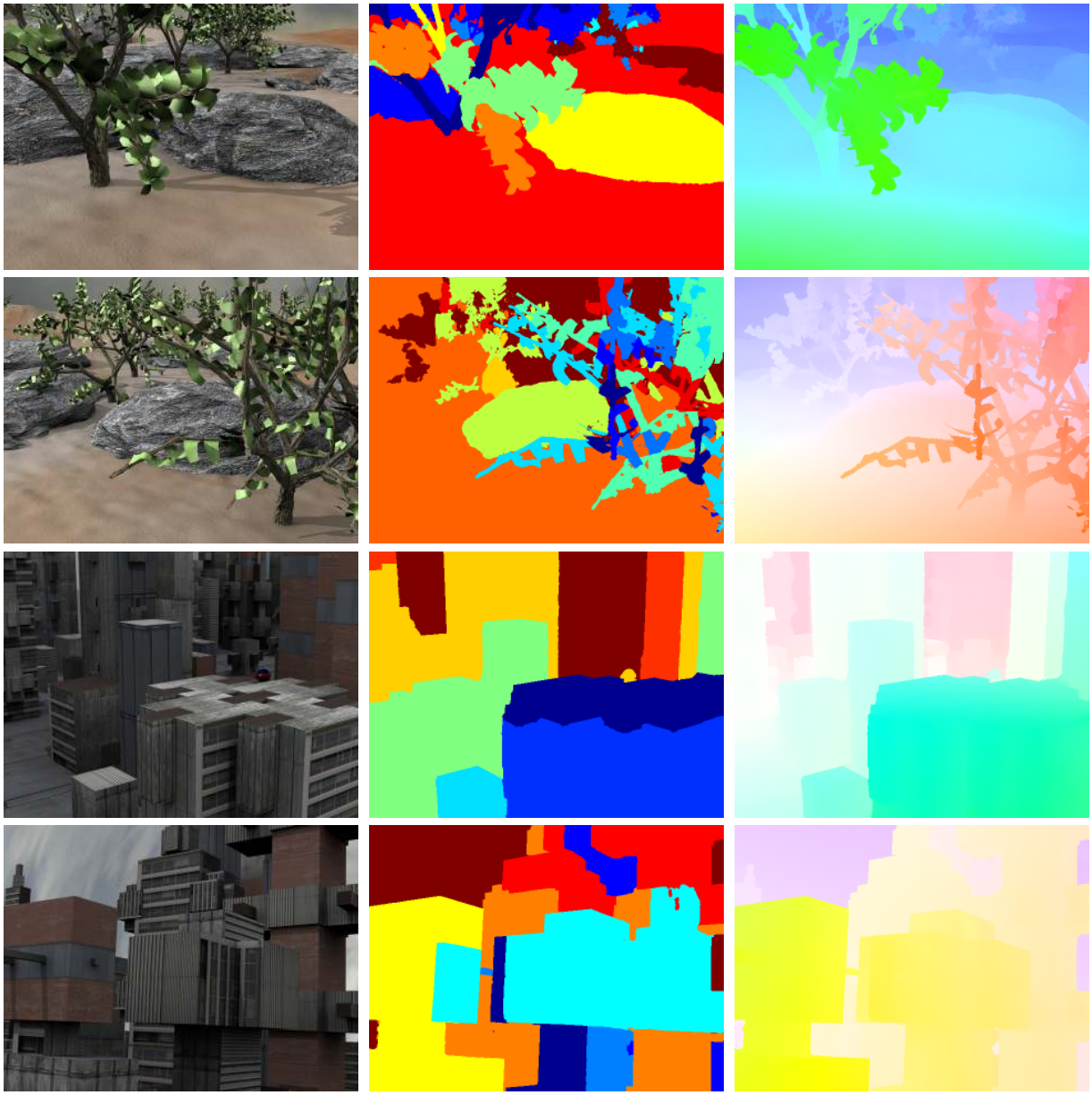
[3] V. Lempitsky, S. Roth, and C. Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. *CVPR*, pp. 1–8, 2008. 4

[4] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. *CVPR*, pp. 2432–2439, 2010. 4

[5] D. Sun, E. Sudderth, and M. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. *NIPS*, pp. 2226–2234, 2010. 6, 11

(a)First frame (b) Layer segmentation (c) Estimated flow field

Figure 7. Estimated flow fields and scene structure on the Middlebury *training* sequences. Left to right: first frame, layer segmentation, and estimated flow field. Top to bottom: "Venus", "Dimetrodon", "Hydrangea", and "RubberWhale".

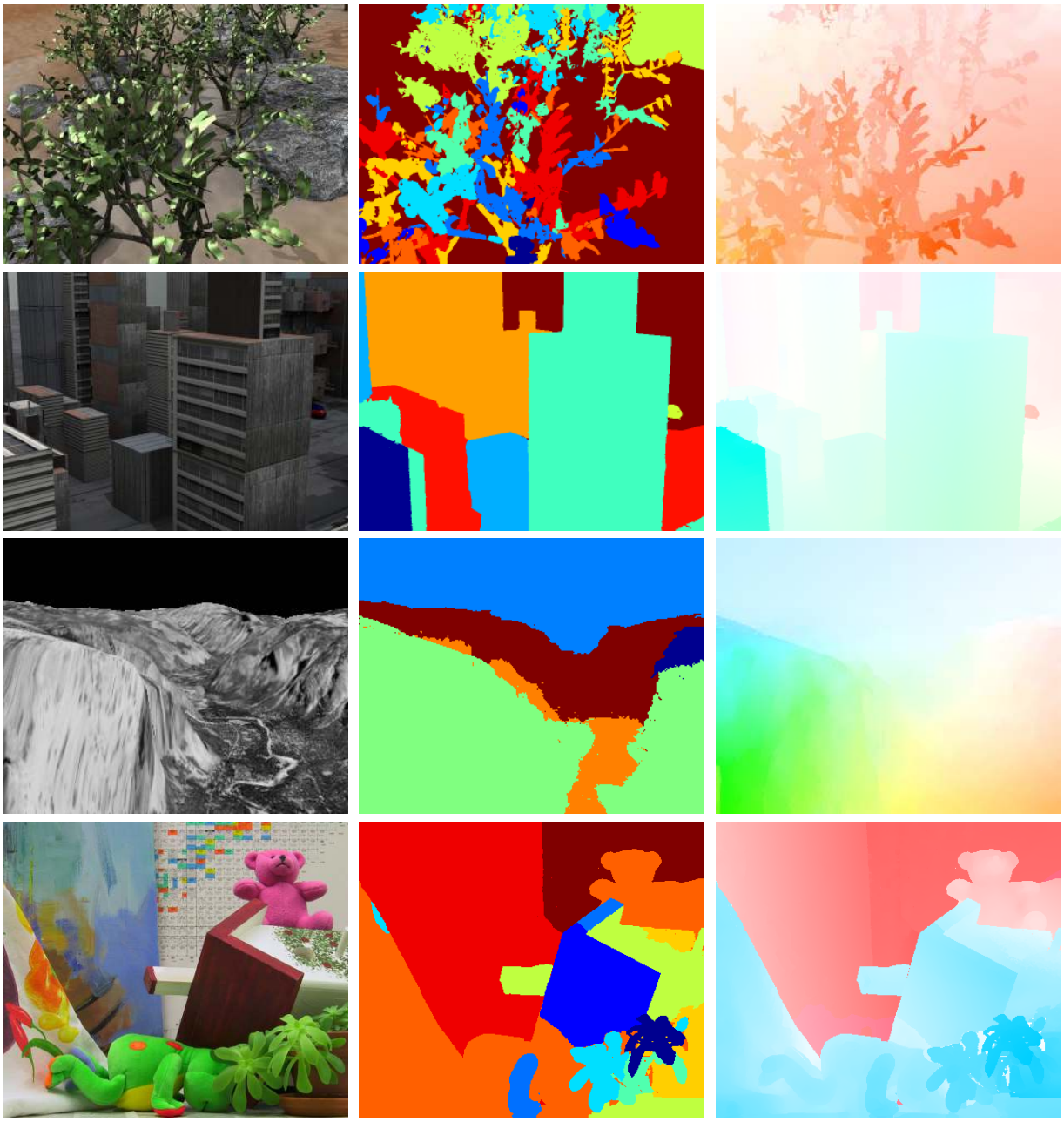


(a)First frame (b) Layer segmentation (c) Estimated flow field

Figure 8. Estimated flow fields and scene structure on the Middlebury *training* sequences. Left to right: first frame, layer segmentation, and estimated flow field. Top to bottom: "Grove2", "Grove3", "Urban2", and "Urban3".

(a)First frame (b) Layer segmentation (c) Estimated flow field

Figure 9. Estimated flow fields and scene structure on the Middlebury *test* sequences. Left to right: first frame, layer segmentation, and estimated flow field. Top to bottom: "Army", "Mequon", "Schefflera", and "Wooden".



(a)First frame (b) Layer segmentation (c) Estimated flow field

Figure 10. Estimated flow fields and scene structure on the Middlebury *test* sequences. Left to right: first frame, layer segmentation, and estimated flow field. Top to bottom: “Grove”, “Urban”, “Yosemite”, and “Teddy”.

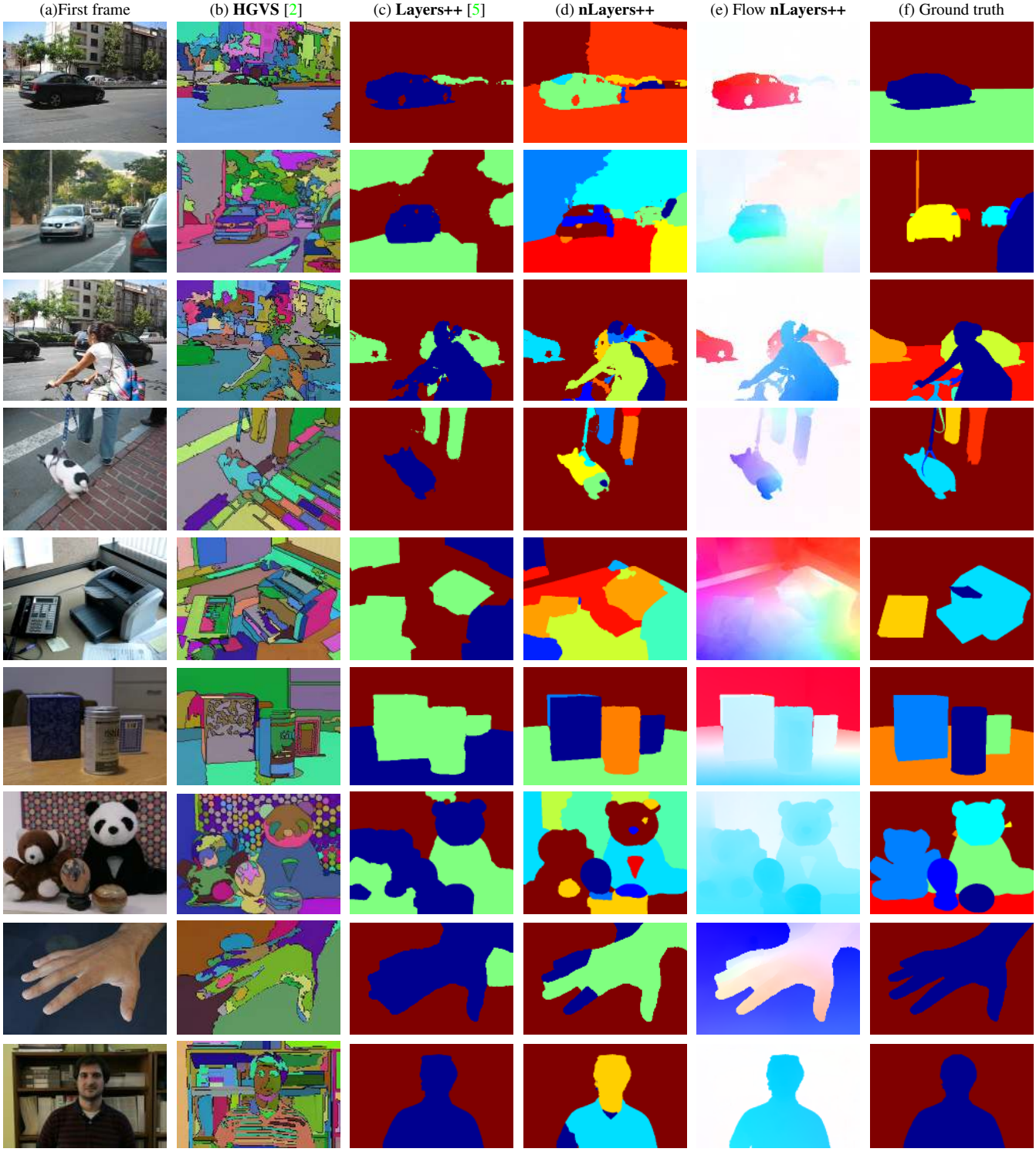


Figure 11. **MIT dataset**. Left to right first frame, segmentation results by **HGVS** [2], **Layers++** [5], **nLayers**, estimated flow field by **nLayers**, and human labeled ground truth. Top to bottom "car", "car2", "car2", "dog", "phone", "table", "toy", "hand", and "person".