$$\sum_{t=1}^{T-1}\sum_{p}\left(\sum_{q\in\mathcal{N}^0_{p,\text{time}}}\phi^0_{\text{time}}(b^p_t,b^q_{t+1})+\sum_{q\in\mathcal{N}^1_{p,\text{time}}}\phi^1_{\text{time}}(b^p_t,b^q_{t+1})\right)+\sum_{t=1}^{T}\sum_{p}\left[\sum_{q\in\mathcal{N}_{p,\text{space}}}\phi_{\text{space}}(b^p_t,b^q_t)+\phi_{\text{affine}}(b^p_t)\right]. \quad (7)$$

that the configuration is unchanged when $b^p_t = 0$, and an alternative layer $\hat{k}$ becomes visible when $b^p_t = 1$. Revealing this new layer may alter all the support functions for the first $\hat{k}$ layers: when $b^p_t = 1$, $g^p_{t\hat{k}}(1) = 1$ and $g^p_{tk}(1) = 0, k < \hat{k}$. In this case, we also set the motion for layer $\hat{k}$ to that of the formerly visible layer ($u^p_{t\hat{k}}(1) = u^{p,\text{old}}_{tk'}$), and the motion for layer k' to its affine mean ($u^p_{tk'}(1) = u^p_{\theta_{tk'}}$).

This segmentation and flow move involves many terms from the overall model, which depend on the binary decision variables as summarized in Eq. (7). The choice of b^p_t influences the flow vector at pixel p, and thus determines which of two candidate pixels it is linked to at the next frame. When $b^p_t = 0$, the temporal neighbors are

$$\mathcal{N}^0_{p,\text{time}} = \{(i + [u^p_{tk}(0)], j + [v^p_{tk}(0)]), 1 \le k \le K\}, \quad (8)$$

and the corresponding potential function will only "fire" when $b^p_t = 0$, so that $\phi^0_{\text{time}}(b^p_t, b^q_{t+1}) =$

$$\Big[\Big(\rho_d(\mathbf{I}^p_t - \mathbf{I}^{q'}_{t+1}) - \lambda_d\Big)s^p_{tk}(b^p_t)s^q_{t+1,k}(b^q_{t+1}) \quad (9)$$
$$+ \lambda_c(1 - \delta(g^p_{tk}(b^p_t), g^q_{t+1,k}(b^q_{t+1})))(1 - \delta(k, K))\Big](1 - b^p_t),$$

which incorporates both the data and temporal terms for the $K - 1$ support functions. We evaluate the warped image $I^{q'}_{t+1}$ at subpixel positions and the visibility mask $s^q_{t+1,k}$ and the warped support function $g^q_{t+1,k}$ at integer positions. Similarly $\phi^1_{\text{time}}(b^p_t, b^q_{t+1})$ is defined to only "fire" when $b^p_t = 1$ (see **Supplemental Material**).

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

$$\phi_{\text{space}}(b^p_t, b^q_t) = \sum_{k=1}^{K-1}\lambda_b w^p_q\big(1 - \delta(g^p_{tk}(b^p_t), g^q_{tk}(b^q_t))\big)$$
$$+ \sum_{k=1}^{K}\lambda_a\rho_{\text{mrf}}\big(u^p_{tk}(b^p_t) - u^q_{tk}(b^q_t)\big). \quad (10)$$

The unary term can be obtained from Eq. (6) as

$$\phi_{\text{affine}}(b^p_t) = \sum_{k=1}^{K}\lambda_a\lambda_{\text{aff}}\rho_{\text{aff}}(u^p_{tk}(b^p_t) - u^p_{\theta_{tk}}). \quad (11)$$

Visibility move. Given the current flow estimate, we decide whether to make a pixel p visible for the selected layer $\hat{k}$ by modifying the previous layer support $\mathbf{g}^{\text{old}}$. When $b^p_t = 0$, all the support functions retain their previous value at p, *i.e.* $g^p_{tk}(0) = g^{p,\text{old}}_{tk}$. When $b^p_t = 1$, we need to adjust the support functions of the first $\hat{k}$ layers so that layer $\hat{k}$ is visible at p, *i.e.* $g^p_{tk}(1) = 0$ if $k < \hat{k}$, and $g^p_{tk}(1) = 1$ 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.

Flow selection move and continuous refinement. Given the current layer assignment, the pixel of each layer can retain its current motion or take the motion of the affine mean flow field. This is a binary segmentation problem similar to the FusionFlow [17] work. The main difference is that our model uses the segmentation information to handle occlusions. After this step, we refine the output flow field by continuous optimization [26] to adaptively change the candidate flow fields for the discrete optimization.

3.3. Layer Number Determination and Depth Order Reasoning

We initialize with an upper bound on the number of layers. During optimization, when a layer has no visible pixels associated with it, we remove it from the solution. The new solution can equally explain the image data, pays no penalty for the removed layer, and so has lower energy. Inferring the depth ordering of layers requires testing all the possible combinations and is computationally prohibitive. We instead use heuristics to reduce the search space. We first order the layers from fast to slow by their average motion. We then perform the moves above to estimate the support functions and the flow fields in both the fast-to-slow and the slow-to-fast ordering. The ordering with the lower energy is further refined as follows. For each pair of neighboring layers, we propose to switch their ordering, and optimize their visibility mask and support functions. If the new solution has a lower energy than its previous one, we accept this new depth ordering and proceed to other pairs (see **Supplemental Material** for the detailed algorithm). In practice, we find that this local greedy search scheme is fairly robust.

4. Experimental Results

We evaluate the proposed layered model on both motion estimation and layer segmentation tasks. Throughout this section, the proposed method is called **nLayers**, since it can automatically determine the number of layers. **Layers++** refers to the continuous method developed in [27] which uses a fixed number of 3 layers. For layer segmentation, we also compare our method to a state-of-the-art, hierarchical graph-based video segmentation algorithm [9], referred to