LSM: Learning Subspace Minimization for Low-level Vision

Chengzhou Tang[1] Lu Yuan[2] Ping Tan[1]

[1]Simon Fraser University [2]Microsoft

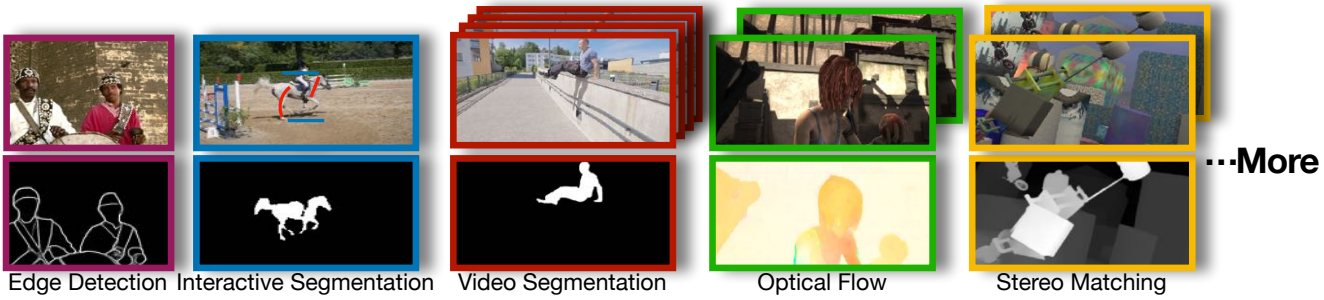


Figure 1: Learning subspace minimization solves various low-level vision tasks using *unified* network structures and parameters.

Abstract

We study the energy minimization problem in low-level vision tasks from a novel perspective. We replace the heuristic regularization term with a learnable subspace constraint, and preserve the data term to exploit domain knowledge derived from the first principle of a task. This learning subspace minimization (LSM) framework unifies the network structures and the parameters for many low-level vision tasks, which allows us to train a single network for multiple tasks simultaneously with completely shared parameters, and even generalizes the trained network to an unseen task as long as its data term can be formulated. We demonstrate our LSM framework on four low-level tasks including interactive image segmentation, video segmentation, stereo matching, and optical flow, and validate the network on various datasets. The experiments show that the proposed LSM generates state-of-the-art results with smaller model size, faster training convergence, and real time inference.

1. Introduction

Many low-level vision tasks (e.g. image segmentation [1, 2, 3], video segmentation [4, 5, 6], stereo matching [7, 8, 9] and optical flow [10, 11, 12]) are formulated as an energy minimization problem:

$$\min_{\boldsymbol{x}} D(\boldsymbol{x}) + R(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ is the desired solution (*e.g.*, a disparity field for stereo matching), and the two terms $D(\boldsymbol{x})$ and $R(\boldsymbol{x})$ are the data term and regularization term respectively. The data term $D(\boldsymbol{x})$ is usually well designed following the first principle of a task, such as the color consistency assumption in optical flow. However, the regularization term $R(\boldsymbol{x})$ is often heuristic. It regularizes $\boldsymbol{x}$ at the *pixel-level* and encourages two *similar* pixels to have similar solution values. Ideally, the regularization term should smooth out noises in $\boldsymbol{x}$ and preserve sharp edges. It has been a major challenge to measure the similarities between pixels, and many different approaches have been proposed, such as the spatial distance in the Tikhonov-Arsenin [13] and the Total Variation (TV) regularization [14, 15], the spatial and color distance in the anisotropic diffusion [16] and the bilteral filter [17], or the distance in a learned feature embedding space [18, 19, 20]. It is still an unsolved problem to design an ideal similarity measurement for efficient and high quality minimization.

We study this minimization problem from a different perspective, where we preserve the data term $D(\boldsymbol{x})$ but replace the heuristic regularization term $R(\boldsymbol{x})$ by a subspace constraint:

$$\min_{\boldsymbol{x}} D(\boldsymbol{x}),\ \text{s.t.}\ \boldsymbol{x} \in \mathcal{V} = \text{span}\{\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots \boldsymbol{v}_K\}, \tag{2}$$

where $\mathcal{V}$ is a K-dimensional subspace, and $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots \boldsymbol{v}_K\}$ is the corresponding basis. In terms of regularization, Eq. (2) introduces an indicator function that $R(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{V}$ and $R(\boldsymbol{x}) = +\infty$ otherwise. But our motivation is quite different: we regularize the problem at the *whole image level* by assuming that the desired solution $\boldsymbol{x}$ is composited of several layers [21, 22, 23], and each basis vector $\boldsymbol{v}_k$ will correspond to one of these layers. Therefore, we are able to represent the solution $\boldsymbol{x}$ as a linear combination of $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots \boldsymbol{v}_K\}$ and solve the combination coefficients, leading to a compact minimization that *not only* is efficient *but also* enables end-to-end training to generate this subspace.

To estimate such a subspace $\mathcal{V}$, we propose the *learning subspace minimization* (LSM) framework that progressively

evolves $\mathcal{V}$ and solves Eq. (2) on a feature pyramid coarse-to-fine. At each pyramid level, we employ a convolutional neural network (CNN) to update $\mathcal{V}$ from both the CNN features of an input image and the derivatives of the data term $D(\boldsymbol{x})$ respect to the intermediate solution $\boldsymbol{x}$. Therefore, $\mathcal{V}$ will be consistently refined when either the image features or the data term captures finer information, which guides $\boldsymbol{x}$ to the optimal solution. Since the generation of $\mathcal{V}$ takes the task-specific data term as *input*, it decouples the task specific characteristics from the subspace generation and unifies the network structure and the parameters for various tasks.

Specifically, when various tasks are unified as a data term minimization, our LSM framework enables joint multi-task learning (*i.e*., train a single network for multiple tasks simultaneously with completely shared parameters), and even makes zero-shot task generalization possible (*i.e*., generalize the trained network as a plug-and-play structure for an unseen task without any parameter modification), as long as its $D(\boldsymbol{x})$ can be formulated.

In the experiments, we implement five low-level vision tasks in an unified paradigm, including interactive image segmentation, video segmentation, stereo matching, optical flow and image diffusion. Our LSM framework has achieved better or comparable results with state-of-the-art fully convolutional methods, and our network structures and parameters can be unified into a compact model, which yields higher efficiency in training and inference. We also demonstrate zero-shot task generalization by leaving one task out for testing and train on the other tasks. All these benefits come from our methodology that integrates domain knowledge (*i.e*., minimizing a data term derived from the first principle) with convolutional neural networks (*i.e*., learn to generate subspace constraint).

2. Related Works

Regularization in Variational Method Many computer vision problems can be formulated as Eq. (1). We only review the continuous settings (i.e. variational method) because it is more relevant to our work and refer readers to [24] for the review about the discrete settings. One of the main focuses is on designing appropriate objective function, especially the regularization term. Rudin and Osher [14] first proposed the TV regularization for image denoising, which has also been proven to be successful for image super-resolution [25], interactive image segmentation [2], stereo matching [7], optical flow [26, 27], multi-view stereo [28], etc. Perona and Malik [16] pioneered to use partial differential equations (PDE) for anisotropic diffusion, which is equivalent to minimizing an energy function with edge-aware regularization [29, 17]. Non-local regularizations [30] have also been proposed for image super-resolution [31], image inpainting [32], optical flow [33], etc, which improves the performance by considering longer range connection between pixels but is usually computational expensive.

Our LSM framework also minimizes an objective function. But we only preserve the data term since it is usually derived from the first principle of a task, and replace the heuristic regularization term to a learned subspace constraint that captures the structure of the desired solution at the whole image context level and enables end-to-end training to boost the performance from data.

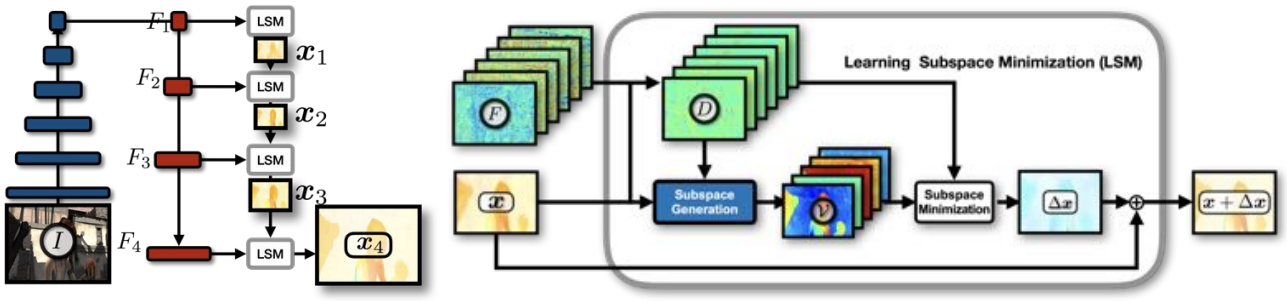
Convolutional Neural Networks Inspired by the success of CNNs in high-level tasks [34, 35, 36], a number of CNN based methods have been proposed for low-level vision tasks. Dong et al. [37] pioneered to use a CNN to upsample overlapped patches for image super-resolution. Zbontar and LeCun [38] and Luo et al. [39] used CNN features to measure image patches' similarity for stereo matching, Xu et al. [40] and Bailer et al. [41] also used CNN based similarity for optical flow. All these methods used CNNs in the patch level, which is computationally expensive and requires post-processing to composite the final result. So more recent works used whole images as inputs. Dosovitskiy et al. [42] used an encoder-decoder structure for optical flow, which is then extended to stereo matching [43] and further evolved in Ilg et al. [44, 45], and other works [46, 47]. Some recent works [48, 49, 50] enabled interactive image segmentation by feeding an image and an user annotation map to CNNs. Meanwhile, CNN based methods [51, 52, 53, 54] have also achieved leading performance for video segmentation.

Our LSM framework also employs CNNs but for different purposes. Instead of predicting the solution directly, we use CNNs to constraint the solution onto a subspace to facilitate the minimization of the data term. The data term is derived from the first principle of each task and decouples the task-specific formulation from the network parameters. Therefore, our framework unifies the network structures as well as parameters for different tasks, and enables zero-shot task generalization, which are difficult for fully CNN based methods. Although some recent works [55, 56] also learn to generate subspace via CNNs, they are designed specifically for 3D reconstruction, which is ad-hoc and unable to generalize to broader low-level vision tasks.

3. Learning Subspace Minimization

3.1. Overview

As illustrated in Fig. 2(a), we first build a feature pyramid $\mathcal{F}$ for *each* image I from a set, where the number of images in a set depends on the task, *e.g*. the interactive segmentation is defined on a single image, the stereo matching and the optical flow are defined on two images, and the video segmentation processes three or more images. The output of the pyramid $\mathcal{F}$ are feature maps in four levels $\{F^1, F^2, F^3, F^4\}$ with strides $\{32, 16, 8, 4\}$ respectively. Please refer to the *supplementary* for the detailed structure of the feature pyramid.



(a) Coarse-to-fine on feature pyramid(s). (b) A single iteration of the learning subspace minimization.

Figure 2: Overview of our learning subspace minimization framework.

At each pyramid level, we define the data term $D(\boldsymbol{x})$ of a task on CNN features (Sec. A) and solve Eq. (2). $D(\boldsymbol{x})$ is approximated using the second-order Taylor expansion at the intermediate solution $\boldsymbol{x}$ and yields the following quadratic minimization problem:

$$\min_{\Delta \boldsymbol{x}} \frac{1}{2} \Delta \boldsymbol{x}^\top \boldsymbol{D} \Delta \boldsymbol{x} + \boldsymbol{d}^\top \Delta \boldsymbol{x}, \tag{3}$$

where $\boldsymbol{D}$ is the matrix that contains the (approximated) second-order derivatives $\frac{\partial^2 D}{\boldsymbol{x}^2}$ of the data term, $\boldsymbol{d}$ is the vector that contains the first-order derivatives $\frac{\partial D}{\partial \boldsymbol{x}}$, and $\Delta \boldsymbol{x}$ is the desired incremental solution. The structure of $\boldsymbol{D}$ is task dependent: it is a diagonal matrix for one-dimensional tasks or block diagonal for multi-dimensional tasks.

To maintain the subspace constraint of Eq. (2), we represent the incremental solution $\Delta \boldsymbol{x}$ as a linear combination of a set of underlying basis vectors, *i.e.* $\Delta \boldsymbol{x} = c_1 \boldsymbol{v}_1 + c_2 \boldsymbol{v}_2 \cdots + c_K \boldsymbol{v}_K$, and then solve the combination coefficients $\boldsymbol{c} = [c_1, c_2 \cdots c_K]^\top$ as:

$$\min_{\boldsymbol{c}} \frac{1}{2} \boldsymbol{c}^\top (\boldsymbol{V}^\top \boldsymbol{D} \boldsymbol{V}) \boldsymbol{c} + (\boldsymbol{d}^\top \boldsymbol{V}) \boldsymbol{c}, \tag{4}$$

where $\boldsymbol{V}$ is a dense matrix, and its columns correspond to the K basis vectors from $\mathcal{V}$. As shown in Fig. 2(b), we generate this $\mathcal{V}$ from the image and the minimization context information (Sec. 3.2), solve minimization with subspace constraint (Sec. 3.3), and move to the next pyramid level after updating the intermediate solution as $\boldsymbol{x} \leftarrow \boldsymbol{x} + \Delta \boldsymbol{x}$.

This formulation is easy and efficient to implement because multiplying the dense matrix $\boldsymbol{V}$ with the (block) diagonal matrix $\boldsymbol{D}$ can be done by column-wise product, yielding a compact K-by-K linear system, which can be solved using direct solver such as Cholesky decomposition [57], instead of iterative solvers such as conjugate gradient descent [58]. Therefore, Eq. (4) is differentiable and supports end-to-end training without unrolling or implicit gradient [59].

3.2. Subspace Generation

Before introducing the network that generates $\mathcal{V}$, we first propose two principles for the subspace generation:

- First, *the image context matters.* Standalone data terms are often insufficient to solve low-level vision problems, because these tasks are usually ill-posed [13], e.g. the well known aperture problem in optical flow [10]. Therefore, it is necessary to consider the image context information. Regularization terms measure the similarities between neighboring pixels by pixel-level information [33, 60, 61]. Instead, we use the whole image context to generate the subspace $\mathcal{V}$, which enforces each basis vector $\boldsymbol{v}_k$ to be spatially smooth except for discontinuities at object boundaries.
- Second, *the minimization context matters.* The objective function (data term) is minimized iteratively. At each iteration, the intermediate solution $\boldsymbol{x}$ is at a different location on the objective function landscape. The minimization depends on the solution space neighborhood, i.e., the local curvature of the objective function decides the direction and magnitude of the incremental solution $\Delta \boldsymbol{x}$ for the minimization. So we incorporate the minimization context along with the image context into subspace generation, which learns to narrow the gap between the estimated solution and the ground truth.

Following these two principles, we learn to generate the subspace $\mathcal{V}$ as illustrated in Fig. 3:

- First, we compute a m-channel image context from the original c-channel feature map F by 1×1 convolution, where $m = c/8$ in our implementation. This step reduces the computation complexity for the following up procedures and balances the impact between the image context and the minimization context.
- Second, we compute a $2m$-channel minimization context. Specifically, we split the c-channel feature map(s) into m groups. Within each group, we evaluate the data term $D(x)$ with the associated feature maps, compute the first-order derivative $\frac{\partial D}{\partial \boldsymbol{x}}$ and the second-order derivatives $\frac{\partial^2 D}{\boldsymbol{x}^2}$, which approximate the objective landscape neighborhood. We concatenate these derivatives to form a $2m$-channel minimization context features.

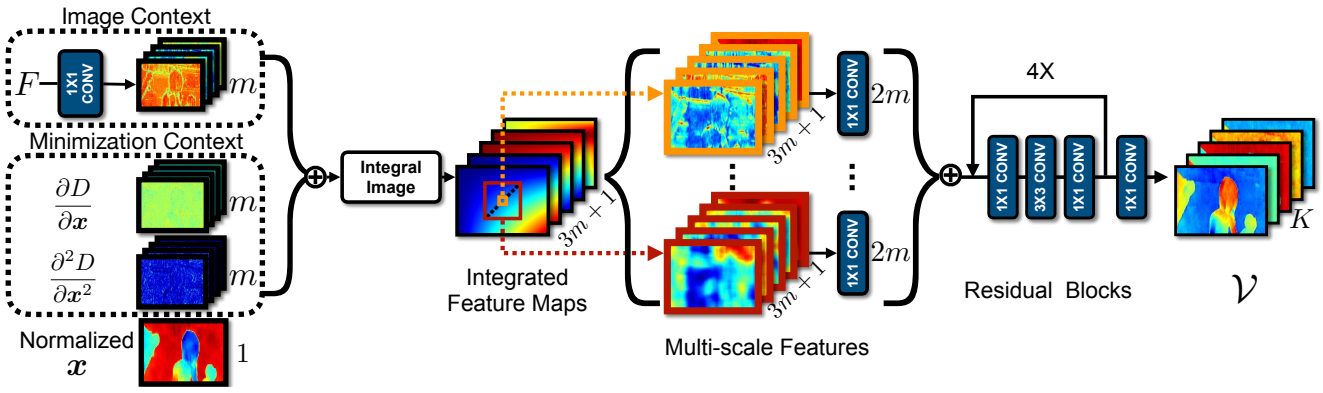


Figure 3: Subspace generation from the image and the minimization context features. The spatial sizes of all feature maps are the same to the F from a feature pyramid level. Integral image is used for the efficient construction of the multi-scale features.

- In the next, we normalize the intermediate solution $\boldsymbol{x}$ with its mean and variance, and concatenate the normalized $\boldsymbol{x}$, the image context, and the minimization context to form a $(3m+1)$-channel input features for subspace generation. To aggregate the context information in multi-scale, we average pool the context features in 4 different kernel sizes without stride, which maintains the spatial size of a feature map. Specifically, we first compute the integral images [62, 63] of the context features and then average neighboring features at each pixel coordinate, which gives better efficiency.
- Finally, we apply a 1×1 convolution to project a feature map to $2m$-channel at each scale individually and concatenate them to get the $8m$-channel multi-scale features. Therefore, we can generate the K-dimensional subspace $\mathcal{V}$ from the multi-scale features via four residual blocks [36] followed by a 1×1 convolution.

3.3. Subspace Minimization

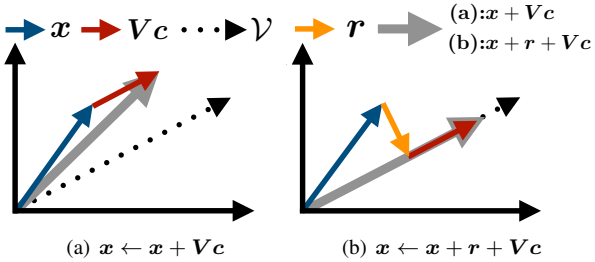


Figure 4: A 2D example where (a): the subspace constraint is violated, i.e. the 2D vector $\boldsymbol{x} + \boldsymbol{V}\boldsymbol{c}$ is not on the 1D subspace $\mathcal{V}$, and (b): the subspace constraint is maintained, i.e. $\boldsymbol{x} + \boldsymbol{r} + \boldsymbol{V}\boldsymbol{c}$ is on $\mathcal{V}$, by considering the residual $\boldsymbol{r}$ between $\boldsymbol{x}$ and its projection on $\mathcal{V}$.

After the subspace $\mathcal{V}$ is generated, we can solve Eq. (4) directly as $\boldsymbol{c} = -(\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{d}$ because $\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V}$ is positive-definite by definition, and update the current intermediate solution as $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{V}\boldsymbol{c}$. However, it will violate the subspace constraint as shown in Fig. 4(a), because the subspace $\mathcal{V}$ is generated progressively, i.e, the current solution $\boldsymbol{x}$ belongs to the subspace from last iteration but is not guaranteed to be on the newly generated $\mathcal{V}$, so is $\boldsymbol{x} + \boldsymbol{V}\boldsymbol{c}$. To address this issue, we propose to project $\boldsymbol{x}$ onto the current subspace $\mathcal{V}$ and reformulate Eq. (4) as follows:

- Denoting $\boldsymbol{P} = \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{V}^\top$ is the projection matrix that projects an arbitrary vector on the the subspace $\mathcal{V}$, we can compute its projection on $\mathcal{V}$ as $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$, and the residual vector from $\boldsymbol{x}$ to $\boldsymbol{x}'$ is $\boldsymbol{r} = (\boldsymbol{P} - \boldsymbol{I})\boldsymbol{x}$.
- Theoretically, we can reevaluate $\boldsymbol{D}$ and $\boldsymbol{d}$ respect to $\boldsymbol{x}'$ and solving Eq. (4), but it requires extra computation. So we reparameterize the incremental solution $\Delta\boldsymbol{x}$ as $\boldsymbol{r} + \boldsymbol{V}\boldsymbol{c}$ and transform Eq. (4) into:

$$\min_{\boldsymbol{c}} \frac{1}{2}(\boldsymbol{r} + \boldsymbol{V}\boldsymbol{c})^\top \boldsymbol{D}(\boldsymbol{r} + \boldsymbol{V}\boldsymbol{c}) + \boldsymbol{d}^\top(\boldsymbol{r} + \boldsymbol{V}\boldsymbol{c}), \quad (5)$$

where we can compute $\boldsymbol{c}$ as,

$$\boldsymbol{c} = -(\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V})^{-1}\boldsymbol{V}^\top(\boldsymbol{d} + \boldsymbol{D}\boldsymbol{r}),$$

without recomputing $\boldsymbol{D}$ and $\boldsymbol{d}$, and update $\boldsymbol{x}$ as $\boldsymbol{x} + \boldsymbol{r} + \boldsymbol{V}\boldsymbol{c}$ as shown in Fig. 4(b).

3.4. Applications

We now show how the proposed LSM framework unifies various low-level vision tasks. We implement five tasks for demonstration, and only introduce the data term for each task. According to the difference of data term formulation, these tasks are divided to three categories.

In the first category, we introduce two binary image labeling tasks: interactive segmentation and video segmentation, both of which share the same formulation as:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \alpha_{\boldsymbol{p}} \|\tau(\boldsymbol{x}_{\boldsymbol{p}}) - 1\|_2^2 + \beta_{\boldsymbol{p}} \|\tau(\boldsymbol{x}_{\boldsymbol{p}}) + 1\|_2^2, \quad (6)$$

where $\boldsymbol{p} = [x, y]$ is a pixel index, τ is an activation function to relax the binary label and constraint $\tau(\boldsymbol{x_p})$ between $(+1, -1)$, and $\alpha_{\boldsymbol{p}}$ and $\beta_{\boldsymbol{p}}$ are the probabilities that $\tau(\boldsymbol{x_p}) = +1$ or -1.

- For **interactive segmentation**, $\tau(\boldsymbol{x_p})$ indicates whether a pixel $\boldsymbol{p}$ is on the foreground object $(+1)$ or background scene (-1), and the corresponding probabilities $\alpha_{\boldsymbol{p}} = \mathcal{G}_f(F_{\boldsymbol{p}})$ and $\beta_{\boldsymbol{p}} = \mathcal{G}_b(F_{\boldsymbol{p}})$, where $\mathcal{G}_{f,b}$ are Gaussian distributions used to model the distribution of features from the foreground scribble points, and the distribution of features from the background scribble points, respectively.
- For **video segmentation**, $\tau(\boldsymbol{x_p})$ indicates whether a pixel $\boldsymbol{p}$ belongs to an previously labeled object $(+1)$ or not (-1), and the corresponding probabilities $\alpha_{\boldsymbol{p}} = \mathcal{P}_f(F_{\boldsymbol{p}})$ and $\beta_{\boldsymbol{p}} = \mathcal{P}_b(F_{\boldsymbol{p}})$ are defined as the average correlation between $\boldsymbol{p}$ and its foreground and background neighbors in previous labeled frames respectively.

In the second category, we introduce two dense correspondence estimation tasks on two images: stereo matching and optical flow, both of which can be formulated as:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \|F_S(\boldsymbol{p} + \boldsymbol{x_p}) - F_T(\boldsymbol{p})\|_2^2, \tag{7}$$

where $\boldsymbol{p} = [x, y]$ is the pixel coordinate in the target (template) image T, and $\boldsymbol{x_p}$ is the warping vector that warps $\boldsymbol{p}$ to $\boldsymbol{p} + \boldsymbol{x_p}$ in the source image S. Similar to the brightness constancy assumption for image channels [10], Eq. (11) assumes that the feature channels F will also be consistent after warping.

- For **stereo matching**, S and T are two images viewing the same scene. Therefore, $\boldsymbol{x_p} = [u, 0]$ only contains horizontal displacement and warps $\boldsymbol{p}$ to $[x + u, y]$ in the target image T.
- For **optical flow**, S and T are two neighboring video frames. Therefore, $\boldsymbol{x_p} = [u, v]$ is the 2D motion vector that warps $\boldsymbol{p}$ to $[x+u, y+v]$ in the S. Since optical flow is a two-dimensional labeling problem compared with stereo matching (one-dimensional, *i.e.* $\boldsymbol{x_p}$ is a scalar) and the two image labeling tasks, we apply Cramer's rule [64] to unify the network structures and parameters of optical flow with others.

The last category is a **image diffusion** [16, 65] task as:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \|\nabla F(\boldsymbol{p})\|_2^2 \|\boldsymbol{x_p} - \boldsymbol{y_p}\|_2^2, \tag{8}$$

where $\nabla F(\boldsymbol{p})$ is the spatial gradient of a feature map, $\boldsymbol{y}$ includes be but not limited to a color image. When $\boldsymbol{y}$ is multi-channel, Eq. (14) can be applied to each channel of $\boldsymbol{y}$ individually to maintain the unified formulation.

As we have seen, the data term formulates various low-level tasks into a one-dimensional or two-dimensional (continuous) image labeling problem, which unifies the network structure and the parameters for various tasks. It not only enables joint multi-task learning in a single network with completely shared parameters, but also allows the network trained on one task to be generalized for another unseen task. Please refer to the the *supplementary* for more implementation details, such as the first-order and the second-order derivatives.

4. Experiments

4.1. Implementation Details

Training Loss Loss design is beyond the scope of this paper, so we use existing losses for all tasks. For *interactive segmentation* and *video segmentation*, we use the Intersection of Union (IoU) loss from Ahmed et al. [66]. For *stereo matching* and *optical flow* we use the end-point-error (EPE) loss as in DispNet [43] and FlowNet [42]. For image diffusion, we apply a 3×3 Sobel operator [67] on the diffused image to detect edges, and use the same classification and regression loss from DeepEdge [68]. Since our solution is estimated coarse-to-fine, we downsample the ground-truth to multiple scales and sum the loss over all scales as in [46].

Hyperparamters We use AdamW optimizer [69] with the default settings where $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate is initialized as 3×10^{-4} and reduced during training using cosine decay [70] without warm restarts. This set of hyperparameters are fixed for all experiments.

Dataset For *interactive segmentation*, we use the PASCAL VOC Semantic Boundaries Dataset [71] for training and the VGG interactive segmentation dataset [3] for testing, and the overlapped 99 images are excluded from the training set. For *video segmentation*, we use the DAVIS-2017 dataset [72] for training and the DAVIS-2016 [73] for testing. For *stereo matching*, we use the FlyingThings3D [43] for both training and testing, and for *optical flow* we use FlyingThings3D for training and Sintel [74] for testing. For *image diffusion (edge detection)*, we use the BSDS500 benchmark [75] for both training and testing.

4.2. Comparison with State-of-the-art

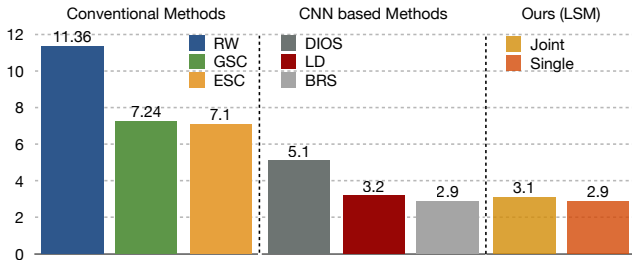
Our framework can be applied to a low-level vision task as long as its first-order and second-order differentiable data term can be formulated. So we first test the multi-task capability of our network. Note that the *whole* network structure and *all* parameters are shared for all tasks, while previous works [76, 77, 78] only share the backbone and use different decoders/heads for different tasks.

We train our model on all five tasks jointly using a workstation with four TITAN-Xp GPUs. For implementation simplicity, we deploy interactive segmentation and image

diffusion on one GPU, and each of the other three tasks on each of the rest three GPUs, and update the network parameters on CPU. The batch size are 6 for interactive segmentation, 6 for video segmentation, 4 for stereo matching and optical flow, and 6 for image diffusion. The training runs for 143.2K iterations. For a fair comparison with other state-of-the-art single-task methods, we also training each task individually and denote the corresponding result as 'Single', while the results of joint training are denoted as 'Joint'.

Interactive Image Segmentation For interactive segmentation, we compare our LSM framework to several conventional methods including ESC and GSC by Gulshan et al. [3], and Random walk [79], as well as recent CNN based methods Deep Object Selection (DIOS) [48], Latent Diversity (LD) [49] and Backpropagation Refinement (BRS) [50]. We randomly sample a few points from the scribbles as inputs for the CNN based methods, since they only supports clicks. We evaluate all methods by the plots required to make IoU greater than 0.85. As shown in Fig. 5(a), our method achieves better results among both recent CNN based methods and the conventional ones.

We also compare with the LD qualitatively when user only interact once. We also subsample scribbles and successively send annotations to LD for a fair comparison. Fig. 5(b) shows that our results are superior than Latent Diversity [49]. It is because the existing CNN based methods only supports spatial distance maps as inputs, which are less precise than scribbles. While our LSM framework supports scribbles by feature distribution estimation and Eq. (14).

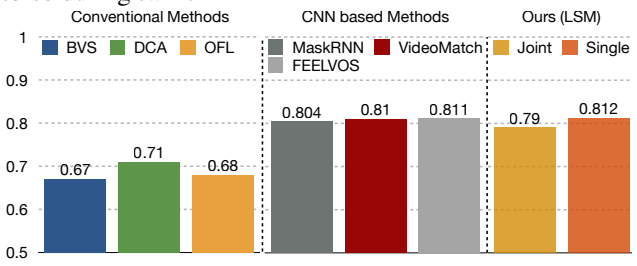


(a) Average plots required to make IoU>0.85.

(b) Our result is superior than LD [49] when user interact only once.

Figure 5: Interactive image segmentation result on VGG interactive segmentation benchmark.

Video Segmentation For video segmentation, we compare our LSM framework to several conventional minimization based methods including BVS [4], OFL [5] and DAC [6], as well as recent CNN based methods that do not require fine-tuning for a fair comparison, which includes MaskRNN [51], VideoMatch [52] and FEELVOS [53]. Fig. 6(a) shows that our LSM framework performs better than conventional method and comparable to CNN based methods. We also show a qualitative comparison to FEELVOS on the challenging dance-twirl sequence. As shown in Fig. 6(b), our LSM framework generates more false positive regions than FEELVOS [53] because the skin and the cloth colors of the dancer and the audiences are similar, but ours is able to track the dancer consistently while FEELVOS lost the dancer's torso during twirl.



(a) Average IoU for video segmentation.

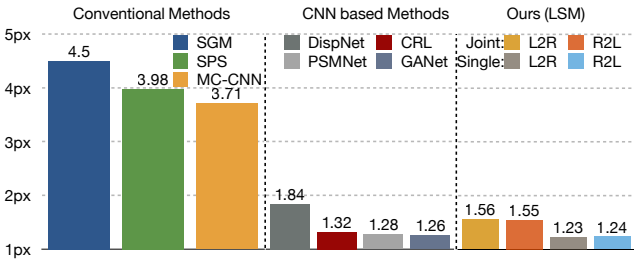
(b) Our result is qualitatively comparable to FEELVOS [53].

Figure 6: Video segmentation result on DAVIS 2016.

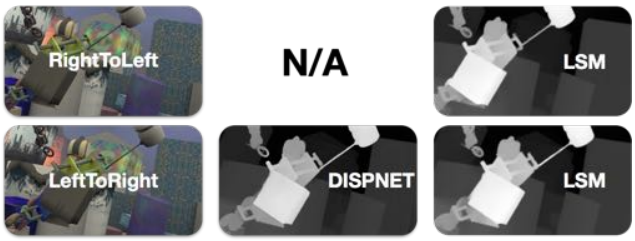
Stereo Matching For stereo matching, we compare our LSM framework with several conventional methods including Semi-Global Matching (SGM) [80], SPS [81] and MC-CNN [82] which uses CNN features *only* for data term evaluation in a Markov Random Fields (MRF), as well as some fully CNN based methods including DispNet [43], CRL [83], PSMNet [47] and GANet [84].

When compared with other CNN based methods, our LSM is comparable for joint training and better for single-task training as shown in Fig. 7(a). As shown in Fig. 7(b), we are able to estimate both the left-to-right and the right-to-left disparities in the same accuracy because we do not assume the direction of the disparity in Eq. (11), while other CNN based methods only deal with left-to-right pairs because of the single directional cost-volume.

Optical Flow For optical flow, we compare our LSM framework with conventional methods including LDOF [85], EpicFlow [86] and PCA-Layers [87] which also adopts a

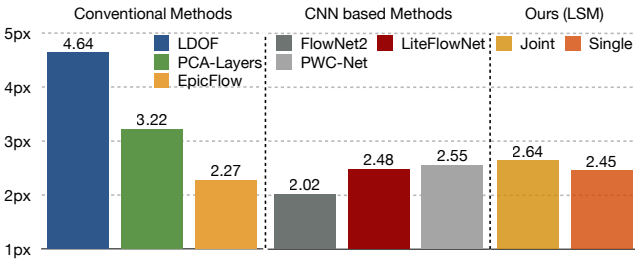


(a) Average End-Point-Error for disparity.

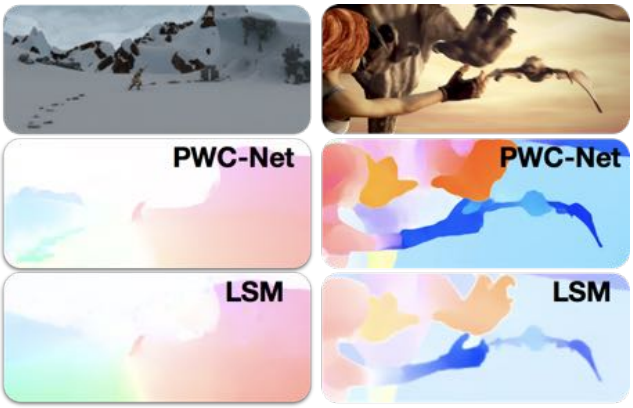


(b) Our LSM supports both left-to-right and right-to-left stereo matching while most of fully CNN based methods only support left-to-right.

Figure 7: Stereo matching results on FlyingThings 3D.



(a) Average End-Point-Error



(b) Our optical flow is comparable to PWC-Net.

Figure 8: Flow trained on Flythings3D and tested on Sintel.

basis representation but the basis is static and specifically learned for optical flow using PCA [88], as well as CNN based methods including LiteFlowNet [89], PWC-Net [46], and FlowNet2-CSS [44] which is a stack of three FlowNets.

As shown in Fig. 8(a), our result are comparable to LiteFlowNet [89] and PWC-Net [46] without refinement sub-net. FlowNet2 is more accurate by stacking networks, which is less efficient, more difficult to train and increases the model size dramatically. Comparing with FlowNet2, our method is $12\times$ smaller in model size, $4\times$ faster in forward pass, and $32\times$ less in training time. The LSM framework is better than LDOF [85] and PCA-Layers [87], and less accurate than EpicFlow [86]. However, the conventional method is usually based on variational approaches and takes 5-20 seconds to run, while our LSM framework takes only 25ms.

Image Diffusion For image diffusion, there are two applications of the diffused image $\boldsymbol{x}$:

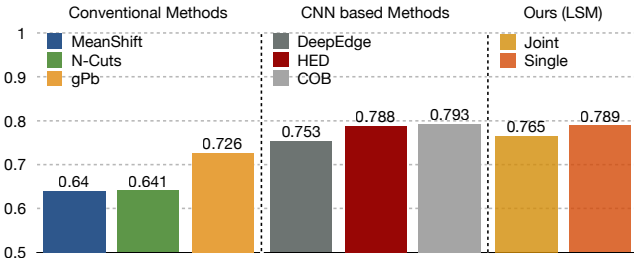
- The first application is **edge detection**. We quantitatively compare the results to several typical conventional methods, including MeanShift [90], Normalized Cuts [91], gPb [92] and Structured Random Forest [93], as well as more recent CNN-based methods, including DeepEdge [68], HED [94] and COB [95]. The comparisons are shown in Fig. 9(a), where our LSM has achieved an F-score (ODS) of 0.789 and is comparable to other CNN based methods. As shown in Fig. 9(b), our edges trade off recall for higher precision while HED tends to hallucinate edges on textured regions.

- We can also use a LSM diffused image $\boldsymbol{x}$ for **super-pixel** segmentation [91, 96, 97, 98]. In Fig. 9(c), we qualitatively compare the SLIC [97] on the original and the diffused RGB images, where the super-pixels on the diffused image are more consistent with object boundaries than the ones on the original RGB images. Another advantage is that the diffused images is still only three- channel, which is more efficient for affinity computation than feature embeddings [99, 100].

4.3. Zero-shot Task Generalization

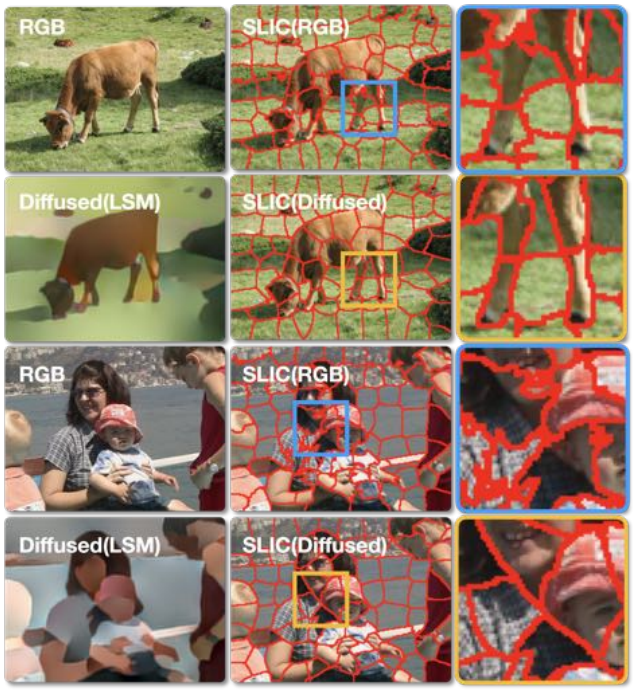
Our LSM framework even generalizes the learned network to unseen tasks. Note that it is different from the zero-shot task transfer [101], where the network parameters are interpolated from existing tasks, and the interpolation coefficients is defined by a correlation matrix during training. In contrast, we fix the learned parameters and do not require any extra information between tasks. To demonstrate this capability, we train the network on three tasks using the same settings as the joint multi-task training, and leave one out for testing.

Stereo Matching The first task left out for testing is stereo matching. Since none of existing CNN based method supports this test, we approximate it by computing optical flow using the PWC-Net [46] on stereo image pairs, and only consider the EPE on the horizontal direction for PWC-Net. The average EPE is 2.47 for our LSM model learned on the other three tasks and tested on stereo matching, which is superior than the 5.29 EPE of PWC-Net as shown in Fig. 10. Note that our LSM framework consistently performs better



(a) Average F-score (higher is better) of edge detection on diffused images.

(b) Our edges are qualitatively more precise than HED.



(c) SLIC super-pixels on the original v.s. the diffused RGB images.

Figure 9: Results of anisotropic image diffusion. (a): The edges detected on the LSM diffused image are quantitatively comparable to other CNN based methods, (b): The detected edges are qualitatively more precise than HED, and (c): the SLIC [97] is more consistent with the object boundaries (*e.g.* cow's legs and human faces) on diffused images.

than conventional methods [80, 81, 82], while PWC-Net is worse than SGM [80].

Optical Flow For optical flow, none of CNN based method

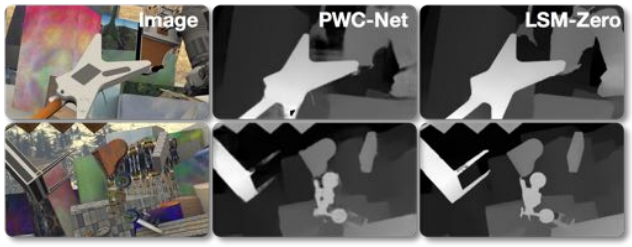


Figure 10: Our zero-shot generalized LSM model performs better than PWC-Net for stereo matching.

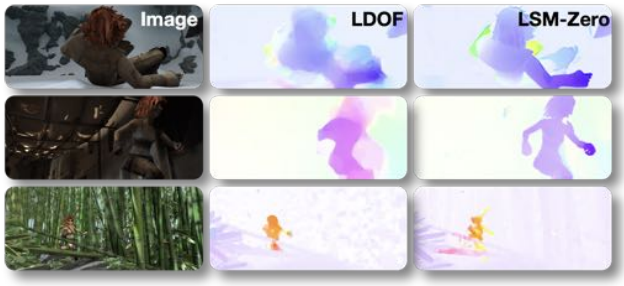


Figure 11: Our zero-shot generalized LSM model performs better than LDOF for optical flow.

supports this zero-shot test, and the average EPE is 4.6 for our LSM model learned on the other three tasks, which is better than LDOF [85]. However, LDOF requires computationally expensive dense HOG [102] feature matching as external input, while our LSM estimates the optical flow efficiently only by minimizing the feature-metric alignment error in Eq. (11). Fig. 11 shows that our zero-shot optical flow maintains the object-aware discontinuities, which indicates that the subspace generator learned from the other three tasks is general, while LDOF generates over-smoothed results because of the $L2$ smoothness regularization term.

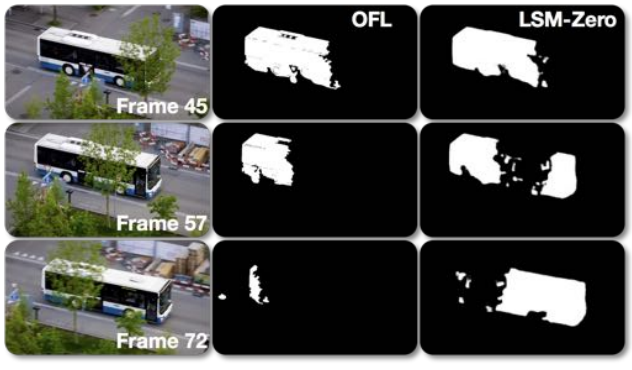


Figure 12: Our zero-shot generalized LSM model is more robust to occlusion than OFL [5] for video segmentation.

Video Segmentation The third task left out for testing is video segmentation. The average IoU is 0.682 for our LSM

model learned on the other tasks and tested on video segmentation, which is comparable to conventional methods such as OFL [5]. However, as shown in Fig. 12, our method is more robust to partial occlusions, while OFL lost tracking of the bus when partially occluded by trees.

Figure 13: Our zero-shot generalized LSM model performs better than GSC [3] for interactive segmentation.

Interactive Segmentation The last task left out for testing is interactive segmentation. When interact only once, the average IoU is 0.802 for our LSM model learned on the other tasks and tested on the interactive segmentation. Which is still superior than the conventional method [3, 79] as shown in Fig. 13.

4.4. Ablation Studies

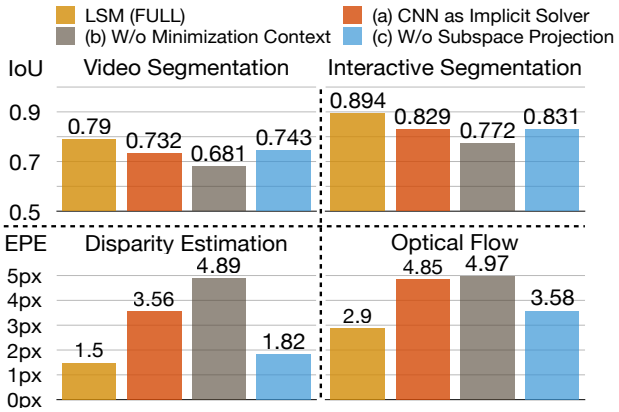


Figure 14: Ablation studies on (a) solving Eq. (3) using CNNs as implicit solver, (b) generating subspace $\mathcal{V}$ without minimization context, and (c) minimization without subspace projection, i.e. directly use Eq. (4).

CNN as Implicit Solver The first question is whether the explicit minimization is necessary, i.e can we use CNN as an implicit solver and predict the solution directly from the image and the minimization context features? To answer this question, we keep the same network structure except the last convolution layer of the subspace generators, i.e. the output of the subspace generator is reduced to one-channel and directly serves as the solution $\boldsymbol{x}$. Then the subspace generator becomes an implicit minimization solver, and the modified network is trained with the same training protocol.

As shown in Fig. 14, without minimization, the interactive segmentation and the video segmentation's get lower IoU while the stereo matching and the optical flow get higher EPE, which indicates the explicit minimization is preferred than learning to minimize via CNNs for our LSM framework.

Without Minimization Context The second question is whether it is necessary to incorporate the minimization context into the subspace generation, i.e can we predict the subspace solely from the image features as in Code-SLAM [55] and BA-Net [56]? To answer this question, we predict the subspace without minimization context and keep the same network structure except the first several convolution layers after the multi-scale context features. The modified network is also trained with the same training protocol in Sec. 4.1.

As shown in Fig. 14, all the four tasks performs significantly worse without the minimization context, which indicates the minimization context is necessary for subspace generation. It is difficult to learn an unified subspace generator solely from image context, because different tasks requires different subspace even on the same image.

Without Subspace Projection Finally, we evaluate the effectiveness of the subspace projection proposed in Sec. 3.3, i.e. minimizing Eq. (4) instead of Eq. (5). We also train the modified network for a fair comparison.

As shown in Fig. 14, the network without the subspace projection performs worse than the original full pipeline, which indicates that maintaining the subspace constraint via projection is necessary not only in theory but also in practice for better performance. It is because, with the subspace projection, the predicted subspace $\mathcal{V}$ is learned to be consistently towards the ground truth solution. In contrast, learning without projection will violate the subspace constraint, and make the minimization less constrained and training more difficult.

5. Conclusions

We propose the learning subspace minimization (LSM) framework to address low-level vision problems that can be formulated as an energy minimization of a data term and a regularization term. We learn convolution neural networks to generate a content-aware subspace constraint to replace the regularization term which is often heuristic and hinders performance At the same time, we exploit the data term and minimize it to solve a low-level task, because the data term is often derived from the first principle of a task and

captures the underlying nature of a problem. This approach nicely combines domain knowledge (*i.e*. minimizing data terms derived from first principles) and the expressive power of CNNs (i.e. learning to predict content-aware subspace constraint). Our LSM framework supports joint multi-task learning with completely shared parameters and also generate state-of-the-art results with much smaller network and faster computation. It even enables zero-shot task generalization, where a trained network can be generalized to unseen tasks. This capability demonstrates our LSM framework can be applied to a wide range of computer vision tasks.

References

[1] Jean Michel Morel and Sergio Solimini. *Variational Methods in Image Segmentation*. 1995. 1

[2] Markus Unger, Thomas Pock, Werner Trobin, Daniel Cremers, and Horst Bischof. Tvseg – interactive total variation based image segmentation. In *British Machine Vision Conference (BMVC)*, 2008. 1, 2

[3] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 1, 5, 6, 9

[4] Nicolas Maerki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral space video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 6

[5] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J. Black. Video segmentation via object flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 6, 8, 9

[6] Emanuel Laude, Jan-Hendrik Lange, Jonas Schüpfer, Csaba Domokos, Laura Leal-Taixé, Frank R. Schmidt, Bjoern Andres, and Daniel Cremers. Discrete-continuous admm for transductive inference in higher-order mrfs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 6

[7] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. Pushing the limits of stereo using variational stereo estimation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 401–407, 2012. 1, 2

[8] R. Ben-Ari and N. Sochen. Variational stereo vision with sharp discontinuities and occlusion handling. In *International Conference on Computer Vision (ICCV)*, pages 1–7, 2007. 1

[9] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. Global solutions of variational models with convex regularization. *SIAM Journal on Image Science*, 3(4):1122–1145, 2010. 1

[10] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence (AI)*, 17(1):185–203, 1981. 1, 3, 5

[11] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 1

[12] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision (IJCV)*, 106(2):115–137, 2014. 1

[13] A. N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems*. Winston and Sons, 1977. 1, 3

[14] L. I. Rudin and S. Osher. Total variation based image restoration with free local constraints. In *International Conference on Image Processing (ICIP)*, pages 31–35, 1994. 1, 2

[15] Vicent Caselles, Antonin Chambolle, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical Foundations and Numerical Methods for Sparse Recovery, De Gruyter, Radon Series Comp. Appl. Math.*, 9:263–340, 2010. 1

[16] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. 1, 2, 5, 14

[17] Sylvain Paris, Pierre Kornprobst, and Jack Tumblin. *Bilateral Filtering*. Now Publishers Inc., 2009. 1, 2

[18] Stefan Roth and Michael J. Black. Fields of experts. *International Journal of Computer Vision (IJCV)*, 82(2):205, 2009. 1

[19] Deqing Sun, Stefan Roth, J. P. Lewis, and Michael J. Black. Learning optical flow. In *European Conference on Computer Vision (ECCV)*, pages 83–97, 2008. 1

[20] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1520–1530, 2017. 1

[21] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 173–178, 1991. 1

[22] J. Y. A. Wang and E. H. Adelson. Layered representation for image sequence coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 221–224, 1993. 1

[23] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 361–366, 1993. 1

[24] Pushmeet Kohli and Carsten Rother. *Higher-order models in Computer Vision*. CRC Press, July 2012. 2

[25] S. D. Babacan, R. Molina, and A. K. Katsaggelos. Total variation super resolution using a variational approach. In *2008 15th IEEE International Conference on Image Processing*, pages 641–644, 2008. 2

[26] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, pages 214–223, 2007. 2

[27] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l1 optical flow. In Daniel Cremers, Bodo Rosenhahn, Alan L. Yuille, and Frank R. Schmidt, editors, *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45, 2009. 2

[28] A. Kuhn, H. Mayer, H. Hirschmüller, and D. Scharstein. A tv prior for high-quality local multi-view stereo reconstruction. In *International Conference on 3D Vision (3DV)*, volume 1, pages 65–72, 2014. 2

[29] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing (TIP)*, 6 (2):298–311, 1997. 2

[30] Gabriel Peyré, Sébastien Bougleux, and Laurent Cohen. Non-local regularization of inverse problems. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *European Conference on Computer Vision (ECCV)*, pages 57–68, 2008. 2

[31] Matan Protter, Michael Elad, Hiroyuki Takeda, and Peyman Milanfar. Generalizing the non-local-means to super-resolution reconstruction. In *IEEE Transactions on Image Processing (TIP)*, page 36, 2009. 2, 16

[32] Pablo Arias, Vicent Caselles, and Guillermo Sapiro. A variational framework for non-local image inpainting. In Daniel Cremers, Yuri Boykov, Andrew Blake, and Frank R. Schmidt, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 345–358, 2009. 2, 16

[33] Philipp Krähenbühl and Vladlen Koltun. Efficient nonlocal regularization for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 356–369, 2012. 2, 3, 16

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 2

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 2

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4

[37] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(2):295–307, 2016. 2

[38] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research (JMLR)*, 17:1–32, 2016. 2

[39] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, 2016. 2

[40] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[41] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(8):1879–1892, 2019. 2

[42] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 5

[43] N. Mayer, E. Ilg, P. Häusser, 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 *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5, 6

[44] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 7, 15

[45] E. Ilg, T. Saikia, M. Keuper, and T. Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[46] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5, 7, 15

[47] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5410–5418, 2018. 2, 6

[48] N. Xu, B. Price, S. Cohen, J. Yang, and T. Huang. Deep interactive object selection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 373–381, 2016. 2, 6

[49] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 577–585, June 2018. 2, 6

[50] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 6

[51] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In *Advances in Neural Information Processing Systems (NIPS)*. 2017. 2, 6

[52] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G. Schwing. Videomatch: Matching based video object segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *European Conference on Computer Vision (ECCV)*, pages 56–73, 2018. 2, 6

[53] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 6

[54] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[55] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2560–2568, 2018. 2, 9

[56] Chengzhou Tang and Ping Tan. BA-net: Dense bundle adjustment networks. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 9

[57] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, 1996. 3

[58] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006. 3

[59] Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012. 3

[60] Xiaofeng Ren. Local grouping for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 3

[61] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4466–4474, 2015. 3

[62] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I, 2001. 4

[63] Kaiming He and Jian Sun. Guided image filtering. In *European Conference on Computer Vision (ECCV)*, 2010. 4

[64] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, second edition, 2002. 5, 14

[65] Joachim Weickert. Anisotropic diffusion in image processing. 1998. 5, 14

[66] F. Ahmed, D. Tarlow, and D. Batra. Optimizing expected intersection-over-union with candidate-constrained crfs. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1850–1858, Dec 2015. 5

[67] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, page 271–272, 1968. 5

[68] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 5, 7

[69] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. 5

[70] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 5

[71] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 5

[72] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 5

[73] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

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

[75] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916. 5

[76] R. Cipolla, Y. Gal, and A. Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, June 2018. 5

[77] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 5

[78] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463, July 2017. 5

[79] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(11):1768–1783, 2006. 6, 9

[80] Amnon Drory, Carsten Haubold, Shai Avidan, and Fred A. Hamprecht. Semi-global matching: A principled derivation in terms of message passing. In Xiaoyi Jiang, Joachim Hornegger, and Reinhard Koch, editors, *Pattern Recognition (PR)*, pages 43–53, 2014. 6, 8

[81] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision (ECCV)*, 2014. 6, 8

[82] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, 2016. 6, 8

[83] Jiahao Pang, Wenxiu Sun, Jimmy SJ. Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017. 6

[84] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[85] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):500–513, 2011. 6, 7, 8

[86] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172, 2015. 6, 7

[87] J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 120–130, 2015. 6, 7

[88] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986. 7

[89] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8981–8989, 2018. 7

[90] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, 2002. 7

[91] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000. 7

[92] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5): 898–916, 2011. 7

[93] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(8):1558–1570, 2015. 7

[94] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015. 7

[95] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Convolutional oriented boundaries. In *European Conference on Computer Vision (ECCV)*, 2016. 7

[96] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, Dec 2009. 7

[97] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012. 7, 8

[98] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. volume 111, 10 2012. 7

[99] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 7

[100] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel samping networks. In *European Conference on Computer Vision (ECCV)*, September 2018. 7

[101] Arghya Pal and Vineeth N Balasubramanian. Zero-shot task transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 7

[102] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2005. 8

[103] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 636–644, 2017. 15

[104] Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *CoRR*, abs/1811.08883, 2018. 15

[105] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2464–2471, 2010. 16

[106] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Transactions of Graphics (TOG)*, 30(4):69:1–69:12, 2011. 16

[107] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2):753–762, 2010. 16

A. Derivatives of Various Data Terms $D(\boldsymbol{x})$

In Sec 3.4, we introduced two categories of tasks. Now, we show the first-order and the (approximated) second-order derivatives of the data terms, which compose the vector $\boldsymbol{d}$ and the (block) diagonal matrix $\boldsymbol{D}$ at each iteration.

Binary Image Labeling Recall that the first category is binary image labeling (interactive segmentation and video segmentation) as:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \alpha_{\boldsymbol{p}} \|\tau(\boldsymbol{x_p}) - 1\|_2^2 + \beta_{\boldsymbol{p}} \|\tau(\boldsymbol{x_p}) + 1\|_2^2, \quad (9)$$

where $\boldsymbol{p} = [x, y]^\top$ is a pixel coordinate, τ is an activation function to relax the binary label $\tau(\boldsymbol{x_p})$ between $(+1, -1)$, and $\alpha_{\boldsymbol{p}}$ and $\beta_{\boldsymbol{p}}$ are the probabilities that $\tau(\boldsymbol{x_p}) = +1$ or -1. Therefore, the first-order and the second-order derivatives at an intermediate solution $\boldsymbol{x}$ are:

$$\begin{aligned} \frac{\partial D}{\partial \boldsymbol{x_p}} &= [(\alpha_{\boldsymbol{p}} + \beta_{\boldsymbol{p}})\tau(\boldsymbol{x_p}) + (\beta_{\boldsymbol{p}} - \alpha_{\boldsymbol{p}})][\frac{\partial \tau(\boldsymbol{x_p})}{\partial \boldsymbol{x_p}}], \\ \frac{\partial^2 D}{\partial \boldsymbol{x_p^2}} &= (\alpha_{\boldsymbol{p}} + \beta_{\boldsymbol{p}})[\frac{\partial \tau(\boldsymbol{x_p})}{\partial \boldsymbol{x_p}}]^2, \end{aligned} \quad (10)$$

where we ignore the scale factor 2 for simplicity, and $\frac{\partial \tau(\boldsymbol{x_p})}{\partial \boldsymbol{x_p}}$ can be $1 - \tau^2(\boldsymbol{x_p})$ for $tanh$ activation function.

Dense Correspondence Estimation The second category is the dense correspondence estimation (stereo matching and optical flow) where the data term is:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \|F_S(\boldsymbol{p} + \boldsymbol{x_p}) - F_T(\boldsymbol{p})\|_2^2. \quad (11)$$

For stereo matching, the derivatives are derived as:

$$\begin{aligned} \frac{\partial D}{\partial \boldsymbol{x_p}} &= \nabla_x F_S(\boldsymbol{p} + \boldsymbol{x_p})^\top [F_S(\boldsymbol{p} + \boldsymbol{x_p}) - F_T(\boldsymbol{p})], \\ \frac{\partial^2 D}{\partial \boldsymbol{x_p^2}} &= \|\nabla_x F_S(\boldsymbol{p} + \boldsymbol{x_p})\|_2^2, \end{aligned} \quad (12)$$

where ∇_x is the gradient operator along the horizontal direction. $\nabla_x F_S(\boldsymbol{p}+\boldsymbol{x_p})$ and $[F_S(\boldsymbol{p}+\boldsymbol{x_p}) - F_T(\boldsymbol{p})]$ are vectors, so $\frac{\partial D}{\partial \boldsymbol{x_p}}$ and $\frac{\partial^2 D}{\partial \boldsymbol{x_p^2}}$ are scalars, which is also a one-dimensional problem and can be unified with the binary image label tasks with the same network and the parameters.

For optical flow, $\boldsymbol{x_p} = [u, v]^\top$ is a 2D vector and the derivatives are:

$$\begin{aligned} \frac{\partial D}{\partial \boldsymbol{x_p}} &= \nabla F_S(\boldsymbol{p} + \boldsymbol{x_p})^\top [F_S(\boldsymbol{p} + \boldsymbol{x_p}) - F_T(\boldsymbol{p})], \\ \frac{\partial^2 D}{\partial \boldsymbol{x_p^2}} &= \nabla F_S(\boldsymbol{p} + \boldsymbol{x_p})^\top \nabla F_S(\boldsymbol{p} + \boldsymbol{x_p}), \end{aligned} \quad (13)$$

where ∇ is the gradient operator along both the horizontal and vertical direction. Therefore, $\frac{\partial D}{\partial \boldsymbol{x_p}}$ is a 2×1 vector, and $\frac{\partial^2 D}{\partial \boldsymbol{x_p^2}}$ is a 2×2 matrix, which makes unification with other one-dimensional tasks difficult. To address this problem, we apply Cramer's rule [64] as follows:

- First, we compute the determinant of $\frac{\partial^2 D}{\partial \boldsymbol{x_p^2}}$ as $det_{\boldsymbol{p}}$.
- Next, we replace the first column of $\frac{\partial^2 D}{\partial \boldsymbol{x_p^2}}$ with $\frac{\partial D}{\partial \boldsymbol{x_p}}$, and denote the determinant of the modified matrix as $det_{\boldsymbol{p}}^x$. Similarly, $det_{\boldsymbol{p}}^y$ is computed by replacing the second column of $\frac{\partial^2 D}{\partial \boldsymbol{x_p^2}}$ with $\frac{\partial D}{\partial \boldsymbol{x_p}}$.
- Finally, we collect $det_{\boldsymbol{p}}^x$ and $det_{\boldsymbol{p}}$ at all pixel locations as the minimization context, concatenate it with the image context to generate the subspace $\mathcal{V}_x$ for the horizontal component of the flow field. Similarly, the $det_{\boldsymbol{p}}^y$ and the $det_{\boldsymbol{p}}$ are collected as the minimization context for the vertical subspace $\mathcal{V}_y$. Thus the subspace generation for optical flow is unified with other one-dimensional tasks by generating the subspace for the horizontal and the vertical components of flow individually.

Image Diffusion The last category is image diffusion [16, 65], and the data term is:

$$D(\boldsymbol{x}) = \sum_{\boldsymbol{p}} \|\nabla F(\boldsymbol{p})\|_2^2 \|\boldsymbol{x_p} - \boldsymbol{y_p}\|_2^2, \quad (14)$$

where $\nabla F(\boldsymbol{p})$ is the spatial gradient of a feature map, $\boldsymbol{y}$ includes be but not limited to a color image, and $\boldsymbol{x}$ is the diffused image. Therefore, the first-order and the second-order derivatives at an intermediate solution $\boldsymbol{x}$ are:

$$\begin{aligned}\frac{\partial D}{\partial \boldsymbol{x_p}} =& \|\nabla F(\boldsymbol{p})\|_2^2(\boldsymbol{x_p} - \boldsymbol{y_p}),\\ \frac{\partial^2 D}{\partial \boldsymbol{x_p^2}} =& \|\nabla F(\boldsymbol{p})\|_2^2.\end{aligned} \tag{15}$$

B. Network Structures

B.1. Feature Pyramid

The feature pyramid learns to extract feature maps that *not only* evaluate the data term $D(\boldsymbol{x})$, *but also* serve as the image context features for subspace generation. We use DRN-22 [103] as the backbone network for efficiency and denote the last residual blocks of conv6, conv5, conv4 and conv3 of DRN-22 as $\mathcal{C} = \{C^1, C^2, C^3, C^4\}$, with channels $\{512, 256, 128, 64\}$ and strides $\{32, 16, 8, 4\}$ respectively. We halve the channels of a feature map C^k by a 1×1 convolution, upsample it by factor of 2 with bilinear interpolation, concatenate the upsampled feature map with C^{k+1} in the next level, and finally apply a 3×3 convolution on the concatenated feature maps to reduce its dimensionality . This procedure is iterated until the finest level, which leads to the final feature pyramid $\mathcal{F} = \{F^1, F^2, F^3, F^4\}$ with the same channels and strides as $\mathcal{C}$.

B.2. Subspace Generator

As introduced in Sec. 3.2, the subspace generator is a stack of residual blocks that generate the subspace $\mathcal{V}$ from the multi-scale context features. The channels for the multi-scale context features introduced in Sec. 3.2 are $\{512, 256, 128, 64\}$ for each pyramid level respectively, and the number of channels are maintain inside the residual blocks. The dimension K of the subspace $\mathcal{V}$, i.e. the output channels of the last convolutional layers of the corresponding subspace generator, are $\{2, 4, 8, 16\}$ at each pyramid level.

B.3. Model Efficiency

Our LSM model is efficient in terms of model size, training time, and inference time, which are contributed by integrating data terms explicitly.

Model Size We implement our LSM framework with the aforementioned settings, which contains about 15M parameters and costs 57.26 MB in memory. As shown in Fig. 15, our LSM model maintains a relatively small model size when compared with other CNN based methods. But our LSM model handles multiple tasks within the same parameters while others are designed specifically for single tasks.

Training Efficiency We train our model with 143.2K iterations for all the experiments, which tasks roughly 20 hours and is relatively faster compared to existing CNN based methods. For example, training FlowNet2 [44] tasks more than 14 days and PWC-Net [46] takes 4.8 days. We initialize the backbone DRN-22 from the ImageNet pre-trained model, which also helps the training converges faster [104].

Inference Efficiency Our LSM framework is also efficient during inference. Since we unify different tasks into a single network, the inference times for various tasks are roughly the same, which consume about 25ms for 512×384 images. The computation is dominated by the feature pyramid construction, the subspace generation and the minimization.

C. Relation to Regularization Term

The Eq. (5) in Sec. 3.3 not only maintains the subspace constraint but also establishes the relation to the conventional energy minimization in Eq. (1), which contains the regularization term and is solved iteratively as:

$$\min_{\Delta \boldsymbol{x}} \frac{1}{2}\Delta \boldsymbol{x}^\top (\boldsymbol{D} + \boldsymbol{L})\Delta \boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{L}\boldsymbol{x})^\top \Delta \boldsymbol{x}, \tag{16}$$

where $\boldsymbol{L}$ contains the second-order derivatives of the regularization term $R(\boldsymbol{x})$ and is called the generalized laplacian filtering matrix, and $\boldsymbol{L}\boldsymbol{x}$ is the first order derivatives of $R(\boldsymbol{x})$.

Proposition 1. *If $\boldsymbol{L} = \boldsymbol{D}(\boldsymbol{P} - \boldsymbol{I})$, and $\Delta \boldsymbol{x}$ is a solution to Eq. (16), its lower dimensional representation (coefficients) $\boldsymbol{c} = (\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{V}^\top \Delta \boldsymbol{x}$ is the solution to Eq. (5).*

Proof. Let $\boldsymbol{L} = \boldsymbol{D}(\boldsymbol{P} - \boldsymbol{I})$,

$$\begin{aligned}(\boldsymbol{D} + \boldsymbol{L})\Delta \boldsymbol{x} =& \\ &= [\boldsymbol{D} + \boldsymbol{D}(\boldsymbol{P} - \boldsymbol{I})]\Delta \boldsymbol{x}\\ &= (\boldsymbol{D} + \boldsymbol{D}\boldsymbol{P} - \boldsymbol{D})\Delta \boldsymbol{x}\\ &= (\boldsymbol{D}\boldsymbol{P})\Delta \boldsymbol{x}\\ &= \boldsymbol{D}\boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{V}^\top \Delta \boldsymbol{x}\\ &= \boldsymbol{D}\boldsymbol{V}[(\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{V}^\top \Delta \boldsymbol{x}]\\ &= \boldsymbol{D}\boldsymbol{V}\boldsymbol{c}.\end{aligned}$$

Since $\Delta \boldsymbol{x}$ is a solution to Eq. (16) which satisfies the second-order optimality condition $(\boldsymbol{D} + \boldsymbol{L})\Delta \boldsymbol{x} = -(\boldsymbol{d} + \boldsymbol{L}\boldsymbol{x})$,

$$\begin{aligned}\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V}\boldsymbol{c} =& \\ &= -\boldsymbol{V}^\top (\boldsymbol{d} + \boldsymbol{L}\boldsymbol{x})\\ &= -\boldsymbol{V}^\top [\boldsymbol{d} + \boldsymbol{D}(\boldsymbol{P} - \boldsymbol{I})\boldsymbol{x}]\\ &= -\boldsymbol{V}^\top (\boldsymbol{d} + \boldsymbol{D}\boldsymbol{r}),\end{aligned}$$

which is the second-order optimality condition for Eq. (5), and $\boldsymbol{c} = -(\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V})^{-1}\boldsymbol{V}^\top (\boldsymbol{d} + \boldsymbol{D}\boldsymbol{r})$. □

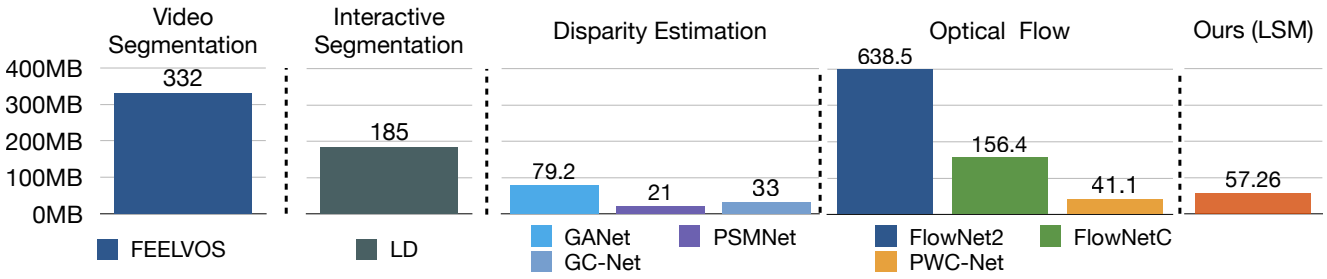


Figure 15: Our LSM model handles multiple tasks in a relatively small model.

The Proposition.1 shows the relation between the proposed LSM framework and conventional variational minimization with regularization term.

In Eq. (16), $\boldsymbol{L}$ is manually defined according to the regularization term. If $R(\boldsymbol{x})$ is the Tikhonov regularization ($L2$ smoothness term), multiplication with $\boldsymbol{L}$ can be implemented efficiently as laplacian filtering, but its performance is poor. More sophisticated may give better performance, but is complicated and expensive, e.g. $\boldsymbol{L}$ is dense for non-local regularization [33, 105, 32, 31] and need to be approximated using high-dimensional filtering [106, 107], but the approximated multiplication still costs more than one minute [33].

In contrast, our LSM framework learns to generate the subspace $\mathcal{V}$ from data, so $\boldsymbol{L} = \boldsymbol{D}(\boldsymbol{P} - \boldsymbol{I})$ adapts to each sample and does not have a fixed form. Since we incorporate the minimization context in subspace generation, $\boldsymbol{L}$ progressively evolves and guides the minimization to a better solution, while conventionally it is defined only based on the image and fixed for all iterations. The LSM framework is also efficient, e.g. we can compute $\boldsymbol{V}^\top \boldsymbol{L}\boldsymbol{x}$ as $(\boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{x} - \boldsymbol{V}^\top \boldsymbol{D}\boldsymbol{x}$, which avoids constructing the dense N-by-N matrix $\boldsymbol{L}$ and the intractable multiplication with it.