Pixel-Level Matching for Video Object Segmentation using Convolutional Neural Networks

Jae Shin Yoon[†‡] Francois Rameau[‡] Junsik Kim[‡] Seokju Lee[‡] Seunghak Shin[‡] In So Kweon[‡]
[†]UMN, Minneapolis, MN [‡]KAIST, South Korea
yoon0074@umn.edu, {frameau, Jskim2, sjlee, shshin}@rcv.kaist.ac.kr, iskweon@kaist.ac.kr

Abstract

We propose a novel video object segmentation algorithm based on pixel-level matching using Convolutional Neural Networks (CNN). Our network aims to distinguish the target area from the background on the basis of the pixel-level similarity between two object units. The proposed network represents a target object using features from different depth layers in order to take advantage of both the spatial details and the category-level semantic information. Furthermore, we propose a feature compression technique that drastically reduces the memory requirements while maintaining the capability of feature representation. Two-stage training (pre-training and fine-tuning) allows our network to handle any target object regardless of its category (even if the object's type does not belong to the pre-training data) or of variations in its appearance through a video sequence. Experiments on large datasets demonstrate the effectiveness of our model - against related methods - in terms of accuracy, speed, and stability. Finally, we introduce the transferability of our network to different domains, such as the infrared data domain.

1. Introduction & Related Works

Video object segmentation refers to the propagation of the mask of an initial object(s) from the first to the last frame of a video sequence. With it, users can determine the pixel-level foreground masks of every image from single key frame supervision. Separating the foreground from the background in a video is a fundamental problem with high applicability to many video based tasks including video summarization, stabilization, retrieval, and scene understanding. For these reasons, video object segmentation has been studied intensively, but it still demonstrates poor results in real world scenarios.

Most recent approaches [5, 23, 29, 1, 25, 28] separate discriminative objects from a background by optimizing an energy equation under various pixel graph relationships.

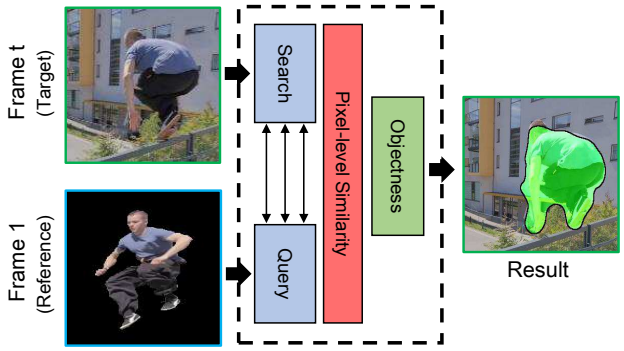

Figure 1. Example of a result of the proposed pixel-level matching network. Here, a completely segmented reference frame is fixed to the query and a sample target frame is fed to the search input. Each colored box is associated with the same color in Fig. 2.

For instance, fully connected graphs have been proposed in [22] to construct a long range spatio-temporal graph structure robust to challenging situations such as occlusion. In another study [9], the higher potential term in a supervoxel cluster unit was used to enforce the steadiness of a graph structure. More recently, non-local graph connections were effectively approximated in the bilateral space [17], which drastically improved the accuracy of segmentation. However, many recent methods are too computationally expensive to deal with long video sequences. They are also greatly affected by cluttered backgrounds, resulting in a drifting effect. Furthermore, many challenges remain partly unsolved, such as large scale variations and dynamic appearance changes. The main reason behind these failure cases is likely poor target appearance representations which do not encompass any semantic level information.

Recently, Convolutional Neural Networks (CNN) have been extensively applied to various vision tasks given their ability to encapsulate semantic information in the object representation task itself. In spite of this advantage, however, only a few attempts [4, 20, 10] have been made to solve video object segmentation problems using a CNN.

Three major causes can be considered: The lack of a training dataset is the primary problem. Indeed, CNN generally requires training with abundant and representative datasets for specific classes. However, it is difficult to account for all types of target object classes given that the class type is always assumed to be undefined for video object segmentation purposes. Many recent benchmarks [7, 13, 21] (including pixel-level ground-truth labeling) provide a partial solution to this problem but nonetheless cannot cover all possible classes. Secondly, training a network exclusively with the initial frame of a video usually leads to over-fitting. If the network is over-fitted to a specific object appearance in the first image, it cannot deal with appearance changes of the target object in the subsequent frames. Finally, the localization of the target object using a CNN is also a challenging issue, especially in relation to pixel-level accuracy. Most CNN structures encode category-level semantic information using features from deep layers generally exploited for object classification. However, they cannot preserve the spatial details of the target object, which are the key components for object localization. For this reason, the idea of combining higher-level features with lower-level ones has attracted much attention in relation to object localization but a unified and elegant method has yet to be proposed.

In order to solve the aforementioned problems, our method is mainly inspired by recent breakthroughs in visual tracking. In a sense, visual tracking shares many common features with video object segmentation (i.e. propagating the initial object mask through a series of images). CNN based visual tracking is also a relatively new trend, but various approaches have already been proposed. We can distinguish two types of visual tracking methods using a CNN, as described below:
1) *Generative model* based tracking aims statistically to describe the appearance of a target and to track the bounding box with the best score compared to previous ones. Some works [16, 24] exploit hierarchical features from different layers to create robust correlation filters. However, these approaches cannot integrate this idea into an end-to-end CNN structure. Nam *et al*. [18] pre-train a network using tracking datasets and fine-tune the model using the initial appearance of the target (in the first image), such that their tracker can be adapted to any type of object. Tao *et al*. [27] validated that matching with Siamese structure can be robust to various challenging scenarios without model updates.
2) *Discriminative tracking* involves the learning of a model that separates distinguishable target objects from the surrounding background. Wang *et al*. [30] effectively deal with the identification problems using the features from conv4 and conv5 layers simultaneously. The criterion used to select the layers, however, causes confusion to the network. In other work [31], a pre-trained network is frequently updated with the appearance of the object to track. For this purpose, the authors reshape the last fully connected layer encoded with the target objectness and calculate the element-wise loss with a conventional regression model.
In consideration of the above tracking approaches, the proposed network embodies the features from lower to higher levels in an end-to-end process. Our learning method also consists of two steps to propose an arbitrary target class adaptation and to cope with appearance variations. The feature extraction part of our network is designed with a Siamese structure to improve the robustness against challenging scenarios. Using the proposed network, our video object segmentation strategy consists of the matching of an object of interest (initialized in the first frame) with subsequent images until the end of a sequence. Therefore, the proposed network must be trained to perform semantic matching between non-successive frames which contain target appearance variations as shown in Fig. 1.

Overall descriptions of the proposed network and the target object mask propagation strategy are described in Fig. 2 and Fig. 3 respectively. The main contributions of this paper are three folded as presented below:

• We propose a novel pixel-level matching network for video object segmentation. Our method shows good computational efficiency as well as higher performance capabilities compared to previous graph based approaches. • We propose a feature *compression* technique that drastically reduces the memory requirements of the network but that also maintains its representation ability. • We experimentally validate the transferability of our network pre-trained on RGB data wiith different domain like infrared data.

2. Proposed Method

If we directly train the proposed network from scratch (random weight parameters) using only the initial frame of a new sequence, the network weights are over-fitted to the first frame object and cannot handle appearance changes. To prevent this, we split the training into two stages. The first involves network pre-training (off-line). In this stage, training involves semantic matching inference in order to deal with appearance variations using a dataset consisting of 300,000 image pairs. The second stage is network fine-tuning (on-line) using the appearance of the object in the first image. This step is indispensable because the target object does not necessarily belong to any pre-trained object class. This makes our network versatile enough to deal with any arbitrary target.

In light of this two-stage learning process, we present specific methods by which to train our pixel-level matching network including its architecture details. We then explain how our model can be applied directly to video object segmentation tasks.

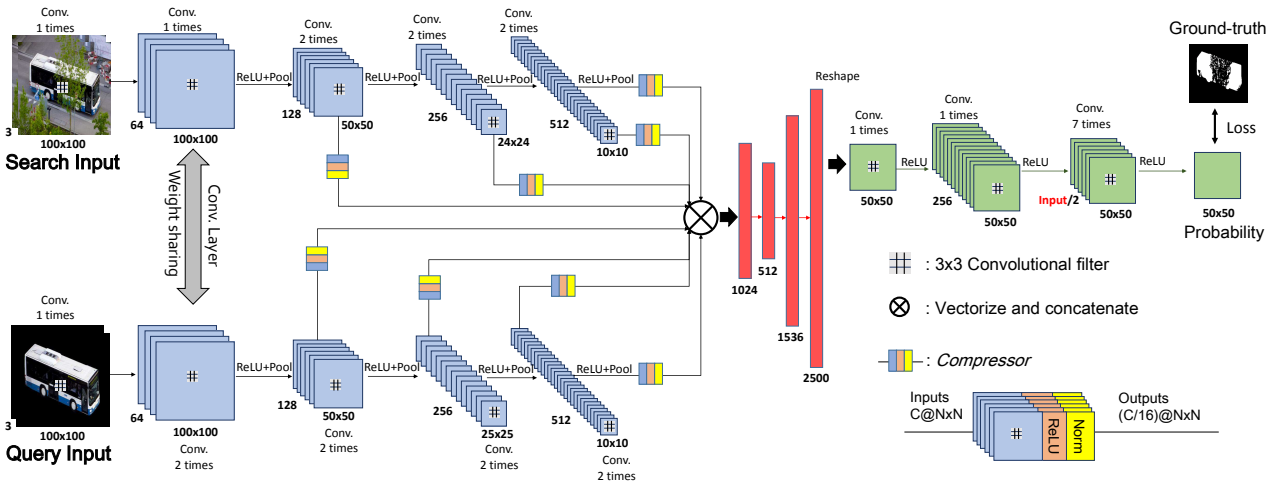

Figure 2. The architecture of the proposed pixel-level matching network. Multi-layered features are extracted from a Siamese structure (blue). Here, all convolutional layers share their weights with mirrored layers including *compressors*. The pixel-level similarity is then encoded via three more hidden fully connected layers (red). Finally, we discriminatively enforce the object coherency through multiple usages of convolutional layers, finally classifying each pixel into the background and the foreground (green). Zero-padding is properly used to fit the output size, while the pooling size is 2×2 with a stride of 2.

2.1. Pretraining Pixel-Level Matching Network

The proposed network is composed of two main parts: pixel-level similarity encoding and target objectness decoding. Our model is pre-trained using 30 video sequences from the *DAVIS* [21] dataset which contains various challenging scenarios. A global description of our network is available in Fig. 2.

Network inputs Since the first part of our network is a Siamese structure, it requires two types of inputs: a reference for the query stream and a target for the search stream. To make our network robust to dynamic appearance variations during the learning process, we choose the target and reference frames randomly (in the same sequence) in order to avoid successive frames (with high similarity).

To generate the query stream dataset, we randomly select 20 reference frames in each training video. For each frame, we crop the bounding box containing the reference object leaving margins around it (25% larger than the original object box size). This region of interest is then completely segmented using the ground-truth label and resized into a 100×100p image patch.

Regarding the search stream data, we randomly select six target frames associated with each reference frame in the same sequence. We then generate the search stream data by cropping and resizing (into 100×100p) multiple bounding boxes at various locations around the target object with three different margins sizes (ρ_1, ρ_2 and ρ_3). By doing this, we can train the model to deal with localization and scale variation. The generated search stream data is further augmented by flipping and rotating. Here, the target object is always fully or partially included in the cropped box (overlapping minimum: 50%). In our experiments, training with hard negative data which does not contain any part of a target object has proved to be rather inefficient.

Pixel-level matching To encode the pixel-wise similarity between the search and query processes, our network initially extracts the features from inputs through several combinations of convolution, max pooling and Rectified Linear Units (ReLU). In the field of visual tracking, one of the major issues is to determine the optimal number of layers to ensure an effective representation of a target object. It was recently found that too many deep layers can be redundant because visual tracking is a binary classification task (foreground or background). Therefore, many works [27, 30, 16, 24] have exploited VGGNet [26] or proposed much lighter structures [31, 18] which demonstrate high performance in terms of accuracy and processing time. Given that our task also belongs to binary classification, we designed our feature extractor similarly. Fig. 2-(blue) shows our feature extraction part. Because our model starts with a Siamese structure, the weights for feature extraction are shared between the search and query streams.

In the fully connected (FC) layers (Fig. 2-(red)), the similarity between the query and search inputs is globally encoded using the extracted features. Here, directly feeding the output features from the last layer in the Siamese structure to the initial FC layer is a good strategy to generate the semantic information of a target object. It can, however, also cause critical localization and identification issues due to the lack of spatial details. On the other hand, the use of lower features alone makes it easy to lose the semantic information. To solve this dilemma, we exploit all of the feature instances from the lower to the higher layers and stack them on the initial FC layer. We then use three more

hidden FC layers to globally combine the multi-level feature instances which encode the similarity between two object units. The size and number of FC layers are similar to that in earlier work [31], which reduces the strong correlation between neighboring pixels to prevent over-fitting problems. A 50×50 matching score table is finally generated by reshaping the last FC layer. However, the use of a large volume of features in the initial FC layer incurs heavy memory requirements which can lead to a critical computational bottleneck. Hence, we compress the output features from each layer (with a data compression ratio of 16) and feed them to the initial FC layer. We call this process *compression* and the associated components *compressors*.

A *compressor* is composed of three types of layers: 3×3 convolution, ReLU, and Local Response Normalization (LRN). The number of convolution filters is sixteen times smaller than the input channel size. The filter weights of a *compressor* are also shared between the query and search stream. Note that the output value of ReLU at each layer shows a different scale due to the unbounded nonlinearity. These unbalanced scales can cause confusion when the feature instances are globally combined in the FC layers. Moreover, pre-normalized features before a loss function induce more effective network convergence as validated in Sec. 3.2. We thus include LRN layer at a *compressor*. Finally, we vectorize and concatenate all of the features from each *compressor* and feed them to the initial FC layer.

The inspiration behind the concept of a *compressor* comes from Wang *et al*. [30]. They fix the number of channels exploited simply by discarding redundant features from the outputs of the conv4 and conv5 layers. Using such a technique, they proved that a target object can be effectively represented without noise. This can, however, cause a critical loss of valuable information depending on the type of data or the complexity of a scene. Therefore, we adaptively compress our features in order to reduce redundancy without losing reliable information.

Objectness decoding The 50×50 matching score table reshaped from the last FC layer encodes the similarity between two input objects. However, direct classification of this matching table at the pixel-level is ineffective due to outliers and inconsistent similarity scores. We thus enforce objectness coherency and reject outliers through several instances of the use of convolutional layers (shaded in green in Fig. 2). Note that one zero padded input is always convoluted by 3×3 filters to maintain the spatial resolution. An example of a clearly separated target object area is available in Fig. 3-(h), and the performance according to the number of decoding layers is validated in Sec. 3.2.

Loss To constraint the network outputs to binary values (background: 1, target: 0) in each pixel, we use the basic element-wise Euclidean distance as a loss function $\mathfrak{L}$. Let the P indexes be the probability of the output and let L denote the value of the ground-truth label. The score is then estimated by

$$\mathfrak{L} = \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} \parallel P(x,y) - L(x,y) \parallel_2, \qquad (1)$$

where $L \in \{0,\ 1\}$, N is the output size, which is set to 50, and (x, y) is the pixel location in the probability map. It should be noted that Sigmoid activation can be used before L2 loss in order to prevent a "strong positive probability" pixel which results in a large penalty. In our case, however, the normalization layer in each *compressor* effectively pre-adjusts the feature scale to [0, 1], resulting in stable loss convergence with the simple L2 function only (Fig. 4-(b)).

2.2. Video Object Segmentation

Fig. 3 shows the pipeline of the proposed video object segmentation method for an arbitrary video. Our method to propagate the initial object mask consists of two parts: candidates sampling and optimal area extraction. Before using our pixel-level matching network, however, it must be fine-tuned (second training stage) during the first frame because the network parameters at the end of the pre-training process are optimized exclusively for the objects in the 30 training videos. Thanks to the supervision of the initial target appearance, the model can adapt to any type of object.

Initial frame fine-tuning In order to generate the fine-tuning data, we use only the first frame of a new video (not utilized for training). For the query input, we completely segment the target object and crop the bounding box around it with small margins (this step assumes a manually labeled image). For the search input, we sample multiple boxes around the target object with different margins (ρ_1, ρ_2, ρ_3) to deal with various scales. It is further augmented by translation and flipping. Here, the target object is always fully or partially included (at least 50% overlapped) for more efficient weight convergence. The input data (about 150 pairs) are finally resized to 100×100p while the corresponding labels are adjusted to 50×50p. We believe that the feature extraction parts are not highly dependent on the appearance of an arbitrary target. Therefore, the weights for the feature extraction parts (the blue boxes in Fig. 2), except for the *compressors*, are fixed, whereas the others are updated.

In the field of visual tracking, many previous works [18, 30, 31] use a frequent model update strategy in the course of a new video sequence. In our case, however, we do not update the network after fine-tuning for the following reasons. First, we need to consider the computational efficiency for video processing. In consideration of this, the model update is particularly heavy (about ten times slower than one single feed forward process). Secondly, the model update strategy has a rather negative effect on video segmentation because we cannot obtain a perfect pixel-level label in the middle of the sequence. Therefore, updating the model using slightly mislabeled data can increase the drift.

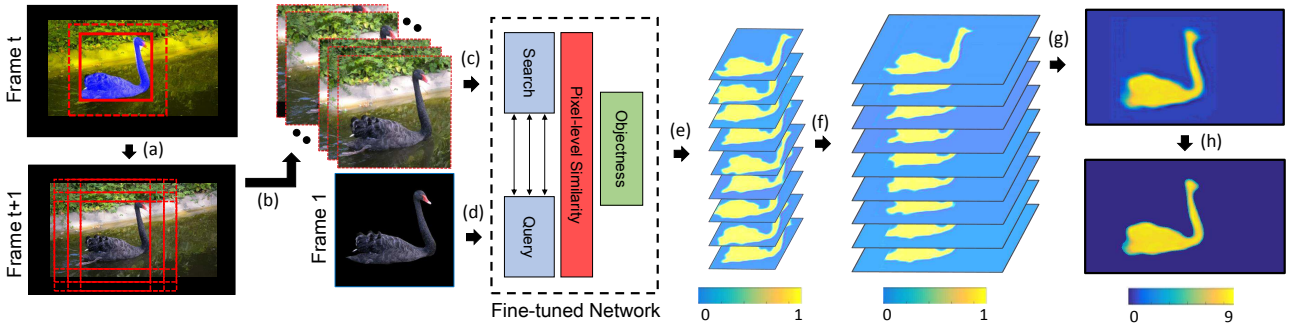

Figure 3. The proposed video object segmentation strategy for an online sequence: (a) Candidates sampling, (b) Image resizing, (c, d) Feeding search and query inputs, (e) Probability maps for each candidate, (f) Size restoration, (g) Stacking, and (h) Thresholding.

Therefore, we match the initial frame with current frame until the end of the sequence without any model update.

Candidates sampling In order to predict reliable target object area in the following frame, we investigate the network responses to multiple candidates and aggregate their information. To do this, we initially sample nine bounding boxes in the next frame in light of the current target object area with certain margins (Fig. 3 - (a)). They are then resized into 100×100p patches to fit the input size of the network (Fig. 3 - (b)). Finally, the patches are fed to the fine-tuned network to obtain the probability maps as shown in Fig. 3 - (c, d, e).

Optimal area extraction Because the size of each probability map is 50×50p, the outputs are reshaped back to the original frame size as described in Fig. 3-(f). We then accumulate all of the response maps aligning the pixel locations. Using the combined information, an optimal target area is finally extracted by thresholding with a pre-defined parameter (Fig. 3-(h)). This strategy is particularly efficient because the output of the network differs slightly according to the location of the box. Therefore, we can extract a more reliable target object area through the aggregation of multiple pixel responses, especially in the edge area. An investigation of the network responses to a larger number of sample candidates may lead to greater performances. In practice, however, we determined that using nine samples is a good compromise between speed and accuracy.

3. Experiments

In this section, we describe implementation details, and we demonstrate the validity of the proposed network structure. Comparative evaluations on three different benchmarks show the effectiveness of the proposed method. Finally, we apply our method to region tracking tasks using infrared data to demonstrate the transferability of our network to a different domain.

3.1. Implementation details

Our CNN pixel-level matching structure is designed and pre-trained using the *Caffe* toolbox. Our video segmentation algorithm (including fine-tuning) is implemented through the MATLAB wrapper of *Caffe*. All of the experiments are conducted on a desktop computer equipped with a 3.60 GHz CPU and a TITAN X graphic card.

We pre-trained our network using a Stochastic Gradient Descent (SGD) method with a learning rate of 10^{-5}. The momentum and weight decay parameters are fixed to 0.9 and 5×10^{-4} respectively. The learning rate is reduced at every 10 epochs, while the parameters of the local size, α and β for the LRN are set to 5, 10^{-4} and 7.5×10^{-1} respectively. For fine-tuning using the initial frame, the learning rate is twice as large (2×10^{-5}) for more rapid convergence, while the other parameters are identical to the pre-training parameters. In our experiments, ρ_1, ρ_2, and ρ_3 are fixed at 10, 30 and 50 respectively. After fine-tuning, online model update does not occur during the sequence.

3.2. Proposed Network Validation

To demonstrate the efficiency of our pixel-level matching network, we conducted a self-structure evaluation on the *DAVIS* [21] benchmark with two different metrics: region similarity (intersection-over-union: IoU) and contour accuracy (calculated from the F-measure score). In order to evaluate the system stability, we also report the mean standard deviation of each metric. Here, the speed for one feed forward task in our network is approximately 8×10^{-3}s, while it takes about 1.5×10^{-1}s per frame to deal with the entire process. The major bottleneck (except for the feed forward time) comes from the image resizing step. In Table 1, the results from the proposed pixel-level matching network and its post-processed results are denoted as PLM and PLM$_P$, respectively. PLM$_P$ is simply obtained by inserting a mask from PLM with the corresponding image into a publicly available weighted median filter [34] code (filter size 5, iteration 3). It makes the PLM results to be sharper and effectively rejects outliers, while it does not have a significant impact on the computational time.

Ablation test for decoding convolutional (DC) layers We conduct an ablation test to find the optimal combina-

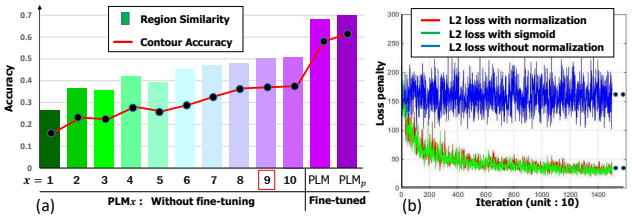

Figure 4. (a) The performance graph according to the number of decoding convolutional layers. The structure PLM_9 is the identical to the PLM. (b) The loss graphs for three different conditions.

tion of DC layers (the green parts in Fig. 2). In this assessment, we add DC layers one by one and stop the layer addition process when the accuracy reaches its maximum as depicted in Fig. 4-(a). The performance (in terms of the average score) for each experimental setting (PLM_1~ PLM_{10}) is evaluated on the *DAVIS* 20 test sequences without fine-tuning. Here, the first DC layer increases the channel size from 1 to $2^{(x-1)}$, where x is the total number of DC layers, while the subsequent DC layers reduce the size of the previous channel by a factor of two. From this graph, we can also demonstrate the effect of fine-tuning (second stage training) by comparing PLM_9 with PLM because they have identical network structures.

Normalization vs. without normalization

Fig. 4-(b) depicts the effectiveness of the normalization layers in the *compressors*. The loss severely fluctuates without normalization, while the normalized features lead to stable loss convergence even without sigmoid activation. We further noted that this convergence tendency is similar to the case when using "sigmoid activation + L2 loss" function.

Use and disuse of *compressor*

As described in Sec. 2.1, owing to the *compressor*, we could shorten the computational time while also boosting the memory efficiency of the network. In order to highlight the advantages offered by the *compression* technique, we attempted to train the network without *compressors*. However, the memory requirement exceeded 12GB which beyond the capacity of our GPU (even for a batch size of 1). This implies that *compressors* are indispensable when training the network (about 3.7GB with a batch size of 32).

Features from multiple layers vs. single layer

To consider the spatial details and semantic information at the same time, we exploit the features from different layers. To underline the importance of this, we compare our network against the same network but using the features from a single layer only denoted as PLM_S. In this case, we directly feed the output of the last convolutional layer (in the Siamese structure) to the initial FC layer. We then train and test the network on the same environment. In Table 1, PLM_S does not work for every sequence due to ineffective pixel-wise object localization, and it is highly affected by cluttered background causing critical drifting effects.

	HVS	NLC	JMP	SEA	BVS	OFL	PLM	PLM_P	PLM_S	PLM_U
blackswan	0.91	0.87	0.93	0.93	**0.94**	**0.94**	0.86	0.89	0.27	0.86
	0.91	0.82	0.94	0.95	**0.96**	**0.98**	0.87	0.89	0.20	0.87
bmx	0.18	0.21	0.23	0.11	0.38	0.14	**0.47**	**0.48**	0.27	0.40
tree	0.28	0.33	0.31	0.13	0.65	0.16	**0.66**	**0.68**	0.43	0.55
break	**0.55**	**0.67**	0.48	0.33	0.50	0.52	0.47	0.48	0.06	0.50
dance	0.47	**0.66**	0.51	0.39	0.49	**0.52**	0.35	0.41	0.08	0.36
camel	**0.87**	0.76	0.64	0.65	0.67	**0.86**	0.65	0.68	0.40	0.62
	0.87	0.72	0.71	0.61	0.70	**0.84**	0.54	0.61	0.20	0.49
car	0.77	0.50	0.72	0.70	0.85	**0.90**	0.86	**0.87**	0.27	0.87
roundabout	0.55	0.25	0.61	0.71	0.62	**0.76**	0.64	**0.71**	0.16	0.68
car	0.70	0.64	0.64	0.77	0.57	**0.84**	0.77	**0.79**	0.43	0.77
shadow	0.59	0.54	0.62	**0.75**	0.47	**0.74**	0.62	0.73	0.32	0.63
cows	0.78	0.88	0.75	0.71	**0.89**	**0.91**	0.81	0.83	0.15	0.79
	0.63	0.80	0.7	0.67	**0.85**	**0.85**	0.71	0.74	0.10	0.65
dance	0.31	0.34	0.44	0.11	0.49	0.53	**0.59**	**0.62**	0.44	0.61
twirl	0.51	0.36	0.52	0.21	0.48	**0.60**	0.50	**0.54**	0.35	0.49
dog	0.72	0.81	0.67	0.58	0.72	**0.89**	0.73	**0.74**	0.29	0.79
	0.63	**0.70**	0.59	0.54	0.59	**0.82**	0.61	0.63	0.15	0.63
drift	0.33	0.32	0.24	0.12	0.03	0.10	**0.73**	**0.71**	0.03	0.61
chicane	0.54	0.31	0.33	0.16	0.07	0.21	**0.79**	**0.79**	0.03	0.60
drift	0.29	0.47	0.61	0.51	0.40	0.33	**0.73**	**0.74**	0.12	0.63
straight	0.26	0.38	0.47	0.50	0.41	0.27	**0.52**	**0.56**	0.05	0.50
goat	0.58	0.01	0.73	0.53	0.66	**0.86**	0.76	**0.77**	0.05	0.74
	0.54	0.13	0.61	0.47	0.58	**0.84**	0.63	**0.69**	0.05	0.60
horse	0.76	**0.83**	0.58	0.63	0.80	**0.86**	0.72	0.77	0.38	0.71
jump	0.80	**0.88**	0.65	0.65	0.80	**0.90**	0.73	0.78	0.24	0.70
kite	0.40	0.45	0.50	0.48	0.42	**0.70**	0.59	**0.64**	0.20	0.25
surf	0.37	0.45	0.31	0.28	**0.64**	**0.49**	0.45	0.45	0.15	0.17
libby	0.55	**0.63**	0.29	0.22	**0.77**	0.55	0.52	0.54	0.23	0.43
	0.64	**0.74**	0.36	0.21	**0.84**	0.61	0.57	0.58	0.21	0.45
motorcross	0.09	0.25	**0.58**	0.38	0.34	**0.60**	0.49	0.51	0.27	0.44
jump	0.13	0.30	**0.54**	0.40	0.37	**0.47**	0.32	0.37	0.06	0.27
paragliding	0.53	0.62	0.59	0.57	**0.64**	**0.63**	0.55	0.57	0.41	0.54
launch	0.20	0.24	0.17	0.18	**0.32**	**0.25**	0.15	0.18	0.11	0.16
parkour	0.24	**0.90**	0.34	0.12	0.75	**0.85**	0.77	0.82	0.05	0.57
	0.32	**0.91**	0.41	0.27	0.67	**0.87**	0.75	0.81	0.08	0.56
scooter	0.62	0.16	0.62	**0.79**	0.33	**0.80**	0.73	0.75	0.09	0.69
black	0.57	0.22	0.53	**0.72**	0.40	**0.73**	0.57	0.64	0.10	0.53
soapbox	0.68	0.63	0.75	**0.78**	**0.79**	0.68	0.72	0.76	0.16	0.70
	0.69	0.65	0.67	**0.75**	**0.76**	0.67	0.54	0.62	0.18	0.49
Average	0.54	0.55	0.56	0.50	0.59	0.67	**0.68**	**0.70**	0.23	0.63
	0.52	0.52	0.53	0.47	0.58	**0.63**	0.58	**0.62**	0.16	0.52
Stability	0.24	0.26	0.18	0.25	0.13	0.25	**0.12**	**0.12**	0.13	0.16
	0.21	0.24	0.18	0.24	0.21	0.25	**0.17**	**0.16**	0.10	0.17
Speed	5s	20s	12s	6s	0.37s	42.2s	**0.15s**	**0.3s**	0.2s	3.85s

Table 1. Performance evaluation on the *DAVIS* [21] benchmark. First: IoU score for region similarity (higher is better). Second: F-measure (higher is better) for contour accuracy. Stability is denoted by the mean standard deviation of each metric (lower is better). At the end of the table, we present the approximate processing time for each method as reported in earlier work [17]. Here, the left part is for a comparative evaluation, while the right one is for the self-structure evaluation. **Red**: best, **blue**: second best. For comparison, the following baseline algorithms are assessed: SEA: SeamSeg [25], JMP: JumpCut [7], NLC: Non-Local Consensus Voting [6], HVS: Efficient Hierarchical Graph-Based Video Segmentation [8], BVS: Bilateral Space Video Segmentation [17], OFL: Video Segmentation via Object Flow [28].

Initial frame matching without a model update vs. successive frames matching with a model updates

To prevent the background drift problem and ensure good computational efficiency, our strategy consists of matching the initially fine-tuned query input with the entire sequence. In order to validate the effectiveness of our strategy, we compare it against successive frame matching with a model update scheme. To do so, the query input for frame $t+1$ is generated using the output mask of the previous frame t. The network model is updated every ten frames through 50

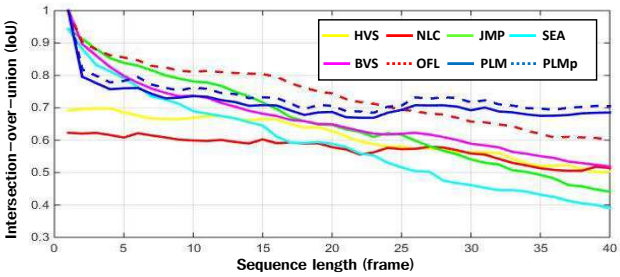

Figure 5. The average IoU graph at each sequence as evaluated on the *DAVIS* [21] benchmark. The performances of most baselines decrease except for NLC as the initial object mask is propagated. Due to the low dependency between successive frames, our method shows stable graph appearance with high accuracy despite the fact that the initial mask reaches the back part of the sequence.

iterations. Here, the model update is done in a manner similar to that in our fine-tuning scheme (described in Sec. 2.2). The performance of the pixel-level matching model with updating is denoted as PLM_U. In Table 1, PLM_U is comparable to PLM, but it never outperforms PLM. This is mainly due to the fact that model updating and matching between two successive frames using continuously drifted outputs lead to rather degenerated results.

3.3. Comparative Evaluation

In our comparative experiments, we measure how much the key frame object mask is precisely propagated through the sequences on four different benchmarks.

DAVIS [21] supplies 30 training datasets and 20 test datasets. This database provides a wide range of challenging scenarios, such as occlusions, dynamic deformations, illumination changes, and scale variations. Note that we did not use any additional datasets to train the network even when testing on different benchmarks.

As described in Table 1, the proposed method generally shows better performance on average compared to all of the other approaches. Among the baselines, OFL is highly comparable but our method is much more efficient in terms of the processing time and system stability which are crucial components in real-world situations. Overall, there are two reasons for these promising results. First, our pixel-level matching network can encompass semantic-level information as well as spatial details. Therefore, our method robustly responds to many challenging scenarios such as occlusions, illumination changes, scale variations, and non-rigid motions. Secondly, unlike previous spatio-temporal graph optimization based approaches [5, 23, 29, 1, 25, 28], our method handles each frame independently. Therefore, a slight mis-segmentation in the current frame does not have induce critical effects on the following frames. The performance gap between the first and last image of the sequences (see Fig. 5) and the stability in Table 1 highlight the robustness of our system to the drifting effect problem. Nonetheless, however, background objects of the same class and with an appearance similar to the target object are highly distractive in our system. Furthermore, the small size of the network output shows difficulty in handling thin objects.

	HSV	FST	DAG	TMF	KEY	NLC	BVS	PLM	PLM_P
birdfall	0.57	0.59	**0.71**	0.62	0.49	**0.74**	0.66	0.64	0.65
cheetah	0.19	0.28	0.40	0.37	0.44	**0.69**	0.10	0.65	**0.65**
girl	0.32	0.73	0.82	0.89	0.88	**0.91**	**0.89**	0.77	0.78
monkeydog	0.68	**0.79**	**0.75**	0.71	0.74	0.78	0.41	0.61	0.72
parachute	0.69	0.91	0.94	0.93	**0.96**	0.94	**0.94**	0.85	0.88
Average	0.49	0.66	0.72	0.70	0.70	**0.81**	0.60	0.70	**0.73**

Table 2. Performance evaluation on the *SegTrack* [13] benchmark using IoU score (higher is better). **Red**: best , **blue**: second best.

	RB	DA	SEA	JMP	PLM	PLM_p
animation	11.9	6.38	6.78	**4.55**	9.38	**5.86**
bball	18.4	8.47	8.89	**3.90**	12.8	**8.04**
bear	4.58	4.48	4.21	**4.00**	8.60	**3.45**
car	**1.76**	5.93	5.08	2.26	4.12	**2.18**
cheetah	31.5	16.6	**7.68**	**8.16**	14.1	11.8
couple	17.5	16.0	23.4	**5.13**	13.1	**9.14**
cup	**5.45**	12.9	9.31	**2.15**	8.63	6.04
dance	56.1	50.8	43.0	**18.7**	31.5	**14.7**
fish	51.8	21.7	25.7	17.5	**9.39**	**7.42**
giraffe	22.0	**11.2**	17.4	**7.40**	19.7	17.4
goat	13.1	13.3	**8.22**	**4.14**	17.8	15.2
hiphop	67.5	51.1	33.7	**14.2**	19.9	**13.6**
horse	8.39	45.1	37.8	**6.80**	11.5	**7.94**
kongfu	40.2	40.8	17.9	**8.00**	9.74	**6.25**
park	11.8	**6.54**	6.91	**5.39**	16.5	10.2
pig	9.22	9.85	10.3	**3.43**	9.09	**5.15**
pot	**2.43**	5.03	2.98	2.95	5.46	**2.66**
skater	38.7	40.8	29.6	22.8	**16.8**	**12.6**
station	8.85	20.9	21.3	9.01	**7.31**	**4.68**
supertramp	129	60.5	57.4	42.9	**30.4**	**20.7**
toy	**1.28**	3.19	2.16	**1.30**	5.07	2.25
tricking	79.4	70.9	35.8	**21.3**	21.9	**15.7**
Average	28.6	23.7	18.8	**9.81**	13.7	**9.55**

Table 3. Errors on the *JumpCut* [7] benchmark using a transfer distance of sixteen (lower is better). **Red**: best , **blue**: second best.

SegTrack [13] To validate the performance of the proposed method more thoroughly, we also conducted experiments on this dataset using the following additional baseline algorithms: FST: Fast Object Segmentation in Unconstrained Video [19], DAG: Video Object Segmentation through spatially accurate and temporally dense extraction of primary object regions [33], TMF: Video segmentation by tracking many figure-ground segments [14], and KEY: Key-segments for video object segmentation [12]. Table 2 summarizes the results. Here, our method shows comparable results but does not always outperform the other approaches. This is mainly due to the fact that our network has not been trained on the same dataset. In addition, the low-quality videos have caused the confusion for our network to distinguish the target object from the background.

JumpCut In order to validate the effectiveness of our model on non-successive scenarios, we use 22 medium resolution videos published by Fan *et al*. [7]. We compare

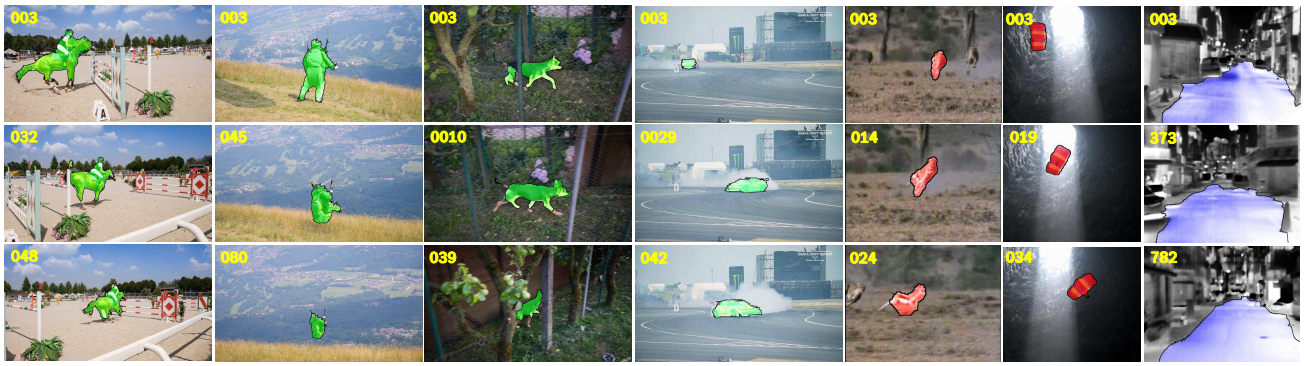

Figure 6. Representative results of the proposed method on challenging scenarios. Each color of the output mask is associated with each benchmark: green, red and blue come from *DAVIS* [21], *SegTrack* [13] and *Thermal-Road* [32] benchmark respectively.

	TD	AlexNet	CN24	FCN	PLM	PLM_p
campus	10.7	42.1	36.4	**10.3**	11.3	**9.89**
mountain	14.0	41.1	21.1	**6.34**	11.4	**10.4**
city	12.6	29.1	28.0	**9.77**	11.95	**11.5**
Average	11.8	39.0	36.7	**9.45**	11.5	**10.6**

Table 4. Error rate (lower is better) evaluation on the *Thermal-Road* [32] benchmark. **Red**: best , **blue**: second best.

our algorithm to the following methods: RB: RotoBrush tool from Adobe AfterEffect [2] based on the SnapCut, and DA: Discontinuity-aware video object cutout [35]. We measured the performance using the same error metric described in [7]. Thus, we investigate the transferred mask from the i^{th} key frame (i= 0, 16,..,96) to the $(i+d)^{th}$ frame. To do this, we fine-tune the network at the key frame and propagate mask skipping for 16 frames (d=16). We then calculate the average error score in each sequence according to the following equation:

$$Err = \frac{100}{N_i} \sum_i \frac{\#\text{ of mislabeled pixels at } (i+d)^{th} \text{ frame}}{\#\text{ of foreground pixels at } (i+d)^{th} \text{ frame}}, \quad (2)$$

where N_i denotes the number of key frames. Overall, the proposed method outperforms all other methods, even without the active contour refinement process introduced in an earlier study [7].

Thermal-Road To demonstrate the applicability of our network to region based tasks such as road tracking, we present an evaluation of the thermal-infrared based road scene data from Yoon *et al*. [32], which contains three different scenarios for a total of approximately 6000 manually annotated images. To generate query and search data on this benchmark, we use the full frame (instead of bounding boxes), as the road occupies nearly the entire image. We conduct a comparison with one hand-crafted feature based method (TD: Thermal-infrared based drivable region detection [32]) and three different CNN based approaches (AlexNet [11], CN24 [3], and FCN [15]). We follow the same error rate metric with [32] calculated by:

$$ErrorRate = \frac{N_{FP} + N_{FN}}{N_P + N_N} \times 100, \quad (3)$$

where N_{FP}, N_{FN}, N_P, and N_N are the number of pixels which are respectively associated with incorrectly detected drivable and non-drivable region, and their ground-truth. As summarized in Table 4, our method outperforms the state-of-the-art hand-crafted feature based approach (TD), and performs second best overall, closely following the FCN approach. Note that, however, we use only an initial frame for fine-tuning our network pre-trained with color images, whereas FCN is fully supervised with the Thermal-Road datasets. From these experiments, we can validate that the proposed network is transferable to a different domain or task using only a single frame. The results also demonstrate that the proposed network is readily trainable with a simple fine-tuning step.

4. Conclusion

In this paper, we proposed a deep learning based video object segmentation algorithm. Our network is composed of encoding and decoding models which are suitable for pixel-level object matching. Two-stage training allows our network to handle appearance variations while also preventing over-fitting problem. We extensively evaluated our method on three widely used benchmark datasets. The obtained results demonstrate that our method performs better than previous related methods in terms of accuracy, speed, and stability. We also verified the importance of using multilayer features and a *compression* technique to make the network compact while maintaining its object representation capability. Finally, we proved the transferability of our network to different domains using the thermal infrared database.

Acknowledgement

This work was supported by the Technology Innovation Program(No. 2017-10069072) funded By the Ministry of Trade, Industry Energy (MOTIE, Korea).

References

[1] V. Badrinarayanan, F. Galasso, and R. Cipolla. Label propagation in video sequences. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3265–3272. IEEE, 2010.

[2] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *ACM Transactions on Graphics (TOG)*, volume 28, page 70. ACM, 2009.

[3] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler. Convolutional patch networks with spatial prior for road detection and urban scene understanding. *arXiv preprint arXiv:1502.06344*, 2015.

[4] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] R. Dondera, V. Morariu, Y. Wang, and L. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *IEEE Winter Conference on Applications of Computer Vision*, pages 784–791. IEEE, 2014.

[6] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, volume 2, page 6, 2014.

[7] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. Jumpcut: non-successive mask transfer and interpolation for video cutout. *ACM Transactions on Graphics (TOG)*, 34(6):195, 2015.

[8] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141–2148. IEEE, 2010.

[9] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *European Conference on Computer Vision*, pages 656–671. Springer, 2014.

[10] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *2011 International Conference on Computer Vision*, pages 1995–2002. IEEE, 2011.

[13] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[14] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2192–2199, 2013.

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3074–3082, 2015.

[17] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 743–751, 2016.

[18] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015.

[19] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1777–1784, 2013.

[20] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017.

[21] F. Perazzi, J. P.-T. B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation.

[22] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3227–3234, 2015.

[23] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *2009 IEEE 12th International Conference on Computer Vision*, pages 779–786. IEEE, 2009.

[24] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, and J. L. M.-H. Yang. Hedged deep tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[25] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 376–383. IEEE, 2014.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. *arXiv preprint arXiv:1605.05863*, 2016.

[28] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 585–594. ACM, 2005.

[30] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.

[31] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015.

[32] J. S. Yoon, K. Park, S. Hwang, N. Kim, Y. Choi, F. Rameau, and I. so Kweon. Thermal-infrared based drivable region de-

tection. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 978–985. IEEE, 2016.

[33] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, 2013.

[34] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter (wmf). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2830–2837, 2014.

[35] F. Zhong, X. Qin, Q. Peng, and X. Meng. Discontinuity-aware video object cutout. *ACM Transactions on Graphics (TOG)*, 31(6):175, 2012.