MovingCables: Moving Cable Segmentation Method and Dataset

Ondřej Holešovský, Radoslav Škoviera, and Václav Hlaváč, *Member, IEEE*

***Abstract*—Manipulating cluttered cables, hoses or ropes is challenging for both robots and humans. Humans often simplify these perceptually challenging tasks by pulling or pushing tangled cables and observing the resulting motions. We propose to use a similar interactive perception principle to aid robotic cable manipulation. A fundamental building block of such an endeavor is a cable motion segmentation method that densely labels moving cable image pixels. This letter presents MovingCables, a moving cable dataset, which we hope will motivate the development and evaluation of cable motion segmentation algorithms. The dataset consists of real-world image sequences automatically annotated with ground truth segmentation masks and optical flow. In addition, we propose a cable motion segmentation method and evaluate its performance on the new dataset.**

***Index Terms*—Data sets for robotic vision, deep learning for visual perception, object detection, segmentation and categorization, cable motion, optical flow.**

I. Introduction

MANIPULATING one-dimensional deformable objects such as cables, hoses or ropes (henceforth referred to as "cables" for brevity), especially when cluttered, is challenging both for humans and robots due to self-occlusions, high-dimensional state space, uniform visual appearance, and complex interaction dynamics. Imagine, for example, that a robot should replace a specific damaged cable in the scene shown in Fig. 1. There are passive computer vision methods [1], [2], [3] for segmenting individual cable instances. However, these methods struggle with occlusions or complex intersections of multiple cables. Novel cable segmentation methods are therefore needed.

Manuscript received 21 February 2024; accepted 2 June 2024. Date of publication 19 June 2024; date of current version 26 June 2024. This letter was recommended for publication by Associate Editor J. Zhu and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by the European Union under the project Robotics and Advanced Industrial Production (reg. no. CZ.02.01.01/00/22_008/0004590) and in part by TAČR project FW08010076. *(Corresponding author: Ondřej Holešovský.)*

Ondřej Holešovský is with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 160 00 Prague, Czech Republic, and also with the Faculty of Electrical Engineering, Czech Technical University in Prague, 166 27 Prague, Czech Republic (e-mail: ondrej.holesovsky@cvut.cz).

Radoslav Škoviera and Václav Hlaváč are with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 160 00 Prague, Czech Republic (e-mail: radoslav.skoviera@cvut.cz; vaclav.hlavac@cvut.cz).

Code, dataset, and visualizations are available at https://github.com/holesond/movingcables and https://doi.org/10.5281/zenodo.11475246.

Digital Object Identifier 10.1109/LRA.2024.3416800

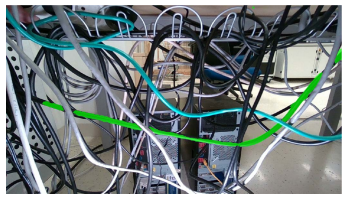
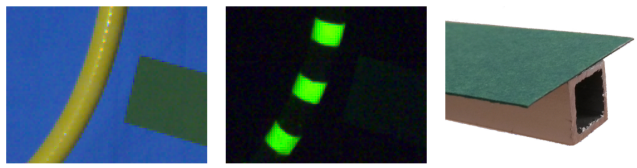


Fig. 1. One of the untidy cables in the scene is moving. Its motion segmentation by MfnProb is in green.



(a) (b) (c)

Fig. 2. Yellow hose, dark green poking stick, blue backdrop. (a) No markers are visible on the hose in white lighting. (b) UV lighting shows the UV fluorescent markers and hides everything else. (c) Detail of the tip of the poking stick.

Our work is inspired by the way humans interactively discover the topology of cluttered cables when trying to untangle them. When a human finds it too hard to visually infer whether two cable segments are directly linked, she grasps and pulls or pushes one of them. The motion visually distinguishes the grasped cable from the clutter. This observation guides us to integrate perception and interaction to aid robotic cable manipulation.

Methods that segment moving cables are an essential building block of the eventually integrated action-perception loop. To test or train such methods, we need a suitable dataset. Creating such a dataset is challenging because we need to obtain not only the cable instance segmentation masks but also the cable motion ground truth. We created an automatically annotated moving cable dataset and a novel method able to segment moving cables.

As our robots are too large to manipulate thin cables gently, we recorded video clips featuring a garden hose being manually pushed by a poking stick. We painted UV fluorescent markers on the hose to facilitate ground truth motion estimation. The UV paint is invisible in regular white light but shines clearly in UV light, see Fig. 2. Marker tracking automatically estimated the ground truth optical flow and chroma key techniques generated cable and poking stick segmentation masks. Finally, we generated video clips featuring multiple overlapping hoses by compositing several single-hose video clips into one.

The contributions of this letter include:

1) MovingCables, the first moving cable segmentation dataset with optical flow and instance segmentation ground truth, is automatically generated by a novel data annotation method.
2) MfnProb, a novel cable motion segmentation algorithm based on an optical flow prediction neural network with probabilistic outputs.
3) An evaluation of five cable motion segmentation algorithms (including MfnProb) on the new dataset demonstrates how the dataset can be used.

The cable motion segmentation methods presented here assume that either the segmentation mask of the arm moving the cables is available or the arm is not visible in the image. In practice, one can obtain the arm mask using, e.g., the arm CAD model and forward kinematics, model-based rigid object segmentation or pose estimation/tracking [4], UV fluorescent markers, or color thresholding (our approach).

Section II discusses the related work. Section III presents the dataset creation process, the automatic data annotation method, and the resulting data's nature. We introduce four cable motion segmentation algorithms based on optical flow prediction neural networks in Section IV. Section V suggests how such algorithms can be evaluated on the dataset, Section VI presents the evaluation results, and Section VII discusses the results and concludes the letter.

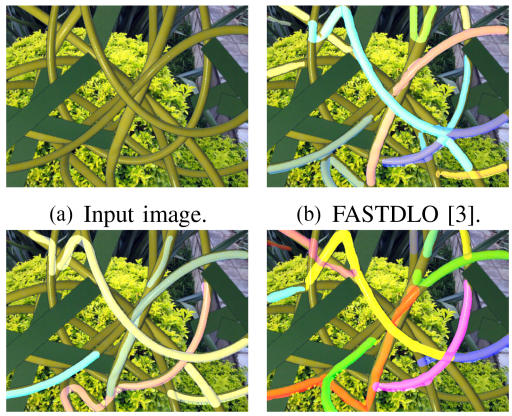
II. Related Work

Existing motion segmentation approaches have been tested mostly on rigid objects. Several methods presented in the literature can segment cables passively from images [1], [2], [3], [5], [6]. Two passive cable segmentation datasets [5], [7] exist, but none with annotated moving cables.

a) Cable perception: Cable segmentation is generally challenging because cables are often of uniform appearance without distinctive features. Several cable detection or segmentation methods in the literature thus relied on simplifying assumptions. Some assumed a single cable was present in the scene [8], [9], others relied on a good cable/background color contrast or on color thresholding to segment the cables from the background [8], [10], [11], [12], [13], [14], [15].

A DeepLabV3+ semantic segmentation neural network can segment wires in an image [5]. Ariadne+ [6] segmented individual wires by processing a superpixel region adjacency graph, taking advantage of the DeepLabV3+ semantic segmentations. An additional TripleNet network predicted the superpixel connectivity scores at wire intersections.

FASTDLO [3] is a recent state-of-the-art passive wire instance segmentation method. It skeletonized each foreground segment predicted by the DeepLabV3+ network to find cable sections, intersections, and endpoints. At each intersection, a similarity neural network paired the neighboring segments with similar color, thickness, and direction estimates. The more recent RT-DLO [2] method replaced FASTDLO's skeletonization with a sparse graph-based approach to handle degraded foreground segments. mBEST [1] found cable instances in skeletonized foreground segments by minimizing the cumulative bending energy of the cables. FASTDLO, RT-DLO, and mBEST may struggle with multiple overlapping cables and severe occlusions, see Fig. 3. We note, however, that scenes involving occlusions or more than two cables at an intersection were outside the scope of mBEST [1]. Zhaole et al. showed that the semantic segmentation networks [3], [5] trained on wire datasets do not generalize well to cables of different textures and color patterns (e.g. ropes) [16]. Their combination of the Segment Anything large vision model with a post-processing method outperformed [3], [5] in segmenting a cable from the background.



(a) Input image. (b) FASTDLO [3]. (c) RT-DLO [2]. (d) mBEST [1].

Fig. 3. Instance segmentation of an image from our dataset.

Deep networks can replace cable state estimation algorithms when task-specific human-labeled training data is available. They can propose interaction keypoints, detect endpoints, classify knots, or refine grasps. Such networks were applied to untangle a multi-cable knot [13], a non-planar knot [17] or a long cable [14]. In [14], an interactive perception algorithm preferred certain manipulation primitives over others when the perception was uncertain. Nevertheless, these approaches assumed that the cables were segmentable from the background by color thresholding. A deep network also helped a robot pick a wiring harness entangled in a pile of wiring harnesses [18]. It predicted the success probability of each available open-loop action given a grasp candidate and a depth image of the scene.

Our work exploits the motion of a cable of interest to simplify the cable perception task, even in complex scenes with multiple overlapping cables and severe occlusions.

b) Interactive segmentation: Interactive perception is the exploitation of forceful robot-environment interactions to simplify and enhance perception [19], [20]. Interactive segmentation [21], a more specific interactive perception skill, interacts with the environment and segments it into a set of movable objects based on the observed motion. It is computationally efficient and requires little prior knowledge about the environment.

Interactive segmentation processes a visual motion signal to segment the moving objects. Options to consider include intensity image differencing with 2D template tracking [21], dense optical flow [22], [23], [24], [25], sparse feature tracking [24], [26], object trackers [26]. Compared to optical flow,

intensity change detection performs poorly when the moved object and the background are of similar color [23] or when multiple objects move [25]. Change detection used together with optical flow improves robustness under strong occlusion, where never-reappearing pixels degrade optical flow [24]. One cannot apply sparse feature tracking to most cables due to their uniform visual appearance.

We have not found any motion segmentation method tested on cluttered cables. To segment cables, we started with a method based on thresholding the magnitude of optical flow predicted by an off-the-shelf neural network [27]. Next, we improved its results by extending it with probabilistic outputs [28] and by retraining it on standard optical flow datasets.

c) Cable datasets: We are not aware of any existing moving cable dataset. Zanella et al. [5] published a static cable dataset for training and evaluating segmentation methods. They took photos of wires on a monochromatic background and randomized it using the chroma key technique. In [7], a human labeled 3D keypoints along a real-world wire using a VR tracker pen. A camera mounted on a robotic arm took images of the wire from different viewpoints. The authors trained semantic and instance segmentation networks on dataset mixtures containing different proportions of synthetic and real-world images. They showed that adding real-world training data improved accuracy at test time.

We propose MovingCables, a novel dataset utilizing UV fluorescent markers to obtain the motion ground truth. UV fluorescence provided the ground truth in datasets for optical flow [29] and the semantic segmentation of rigid and deformable objects [30], [31]. Baker et al. [29] painted fluorescent speckles onto several objects, including clothes. They switched between visible and UV light to record images with and without the speckles. The Lucas-Kanade algorithm estimated the ground truth optical flow even for low-textured objects thanks to the speckles. Instead of relying on speckles, we opt for stripe markers to obtain uninterrupted marker trajectories extending across the entire video recording.

III. MovingCables Dataset

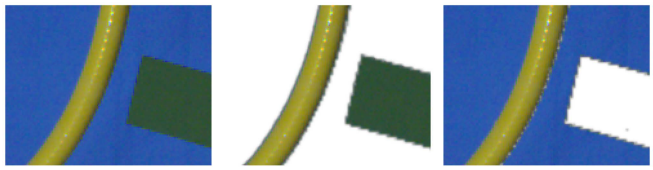
Here we present the dataset creation process, the automatic data annotation method, and the nature of the resulting data. We started by recording the video clips of a single hose with a blue screen in the background (Section III-A). Chroma key segmentation and UV fluorescent marker tracking automatically annotated these images with optical flow and segmentation masks (Section III-B). Finally, we composited multiple recorded single-hose clips and various background images (Section III-B) to obtain the final composed dataset consisting of clips showing multiple overlapping hoses (Section III-C).

A. Raw Data Recording

A Basler ace aCA640-750uc camera with a 6 mm lens recorded the moving cable scene. A frame standing in front of the camera held the two endpoints of a plain yellow garden hose. We placed a blue screen in the background.

(a) (b) (c)

Fig. 4. (a) Vertical white lights light the blue screen background from the left and right. (b) Shining UV light strips with white light turned on. (c) Only UV lights turned on.



(a) (b) (c)

Fig. 5. (a) Blue screen, yellow hose, green poking stick. (b) Transparent background. (c) Transparent poking stick.

The poking stick, see Fig. 2(c), was a long thin aluminum bar with dark green cardboard attached to one of its faces. We ensured the cardboard faced the camera when recording to keep the aluminum bar hidden.

The UV fluorescent stripe markers in Fig. 2(b) are cylinder shells painted on a cable in regular intervals with transparent UV fluorescent paint (UV-elements Invisible Glow Lacquer green[1]).

White LED strips two meters tall lit the background blue screen from the sides, see Fig. 4(a). Another set of vertical UV LED strips (370 nm wavelength) illuminated the cables, see Fig. 4(b), (c). Solid-state relays turned the white and UV LED strips rapidly on and off. White LED strips could also illuminate the cables in the foreground with visible light. Instead, we used high-power white SMD LEDs and a custom LED driver with a digital PWM/enable control input.

The camera recorded the scene at 640×480 pixels, 120 FPS. Its digital trigger output signal emitted at the start of every exposure controlled the lights. A UV-lit image followed each white-lit image taken by the camera so that the white-lit image sequence was recorded at 60 FPS. We recorded one video clip per one poking interaction. Each clip is 10 seconds long and contains ca. 1201 images. The raw recorded dataset consists of 177 clips and 212 581 images.

B. Post-Processing

We post-process the recorded clips in two stages. The first stage performs chroma keying, marker detection, marker tracing, and optical flow ground truth computation. Foreground-background compositing and data augmentation run separately in the second stage.

a) Chroma keying: We use chroma keying to key the blue screen and the green poking stick, see Fig. 5. Chroma keying

[1]https://www.uv-elements.de/shop/en/Invisible-Glow-Lacquer-50ml-green

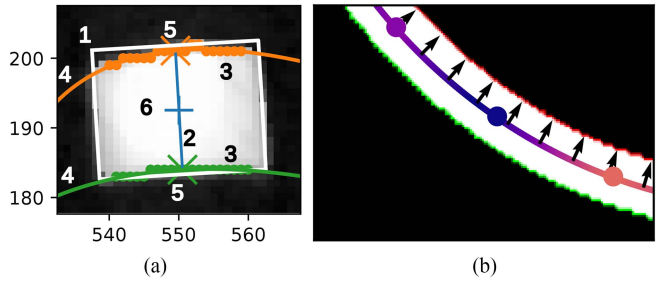


Fig. 6. (a) Marker center point detection. 1) Fit the minimum area rotated bounding rectangle to the blob. 2) Rectangle (and marker) center line. 3) Scan along lines parallel to the center line. Find the endpoints of the line segments entirely within the blob. 4) Fit a parabola to each set of endpoints. Use orthogonal distance regression (ODR). 5) Intersect each parabola with the center line to estimate the central segment. 6) The central segment center is the marker center. (b) Interpolating optical flow along a cable backbone (middle curve). The cable segmentation mask is white, its contour lines are red and green. The dots represent marker centers, their colors indicate the optical flow magnitude. Black arrows show the unit normal vectors of the backbone spline.

segments the key color image regions by thresholding the red, green and blue color channels.

b) Marker detection and tracing: The stripe marker detector detects the UV fluorescent markers in the images of each clip. We ensure during recording that most markers are visible in all the frames of a clip and the poking stick occludes none of them. As the detector does not measure the marker depth, the cables should ideally move in a plane parallel to the image plane. Nearest neighbor marker tracking finds the traces of individual markers. Position interpolation of the traces estimates the marker position in the white-lit images. Given the complete marker traces, one can compute marker velocity or displacement for any image pair.

The marker detector extracts individual marker blobs by thresholding a UV-lit image using a fixed intensity threshold. It then locates the center point of the marker blob, see Fig. 6(a).

c) Optical flow ground truth: Optical flow is an independent per-pixel estimate of motion between two images [32]. Given the current image I_j sampled at $x_i \in \mathbb{R}^2$ discrete pixel locations, optical flow vectors $\phi_i \in \mathbb{R}^2$ estimate the location of these pixels in a reference image I_1. The optical flow minimizes the brightness or color difference between corresponding pixels summed over all the pixel locations of the current image, $\sum_i [I_1(x_i + \phi_i) - I_j(x_i)]^2$.

We provide two types of flow ground truth: full optical flow and "normal flow". Sufficiently textured cables allow full optical flow estimation. "Normal flow" is relevant for textureless cables that only exhibit motion at their boundaries. It is the normal projection of the optical flow vector on the cable boundary normal unit vector. Both ground truths neglect motions caused by a cable rotating around its axis.

In the recordings, the cable never crosses itself and its endpoints are outside the image. Given a cable segmentation mask (a binary image) and marker traces, interpolation estimates the ground truth flow for each cable pixel.

Thresholding the background-foreground alpha matte yields the foreground mask, and thresholding the poking stick alpha matte yields the poking stick mask. We dilate the poking stick mask by two pixels to ensure that (almost) all poking stick boundary pixels are segmented. The cable segmentation mask is the foreground mask with the poking stick mask pixels removed (set to zero).

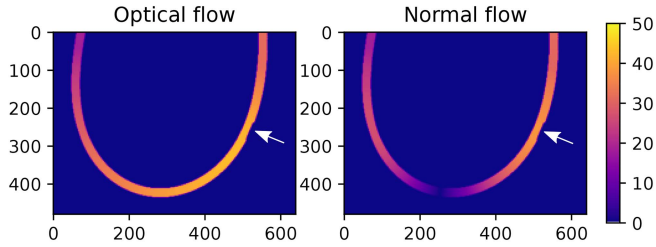
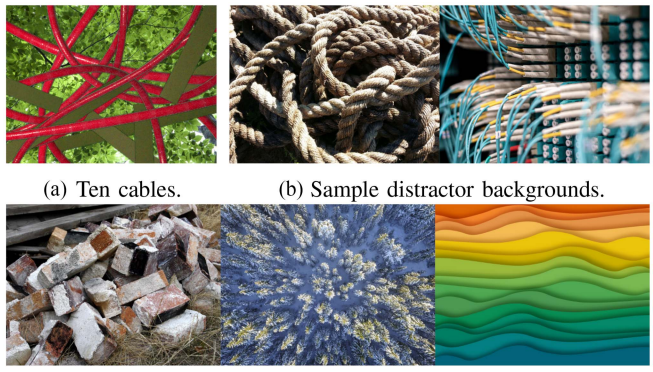


Fig. 7. Sample ground truth optical and normal flow magnitude in pixels when poking the cable at the right side towards the left as marked by the white arrow.



(a) Ten cables. (b) Sample distractor backgrounds.

(c) Sample clutter backgrounds.

Fig. 8. A sample composition (a) and backgrounds (b, c).

The interpolation process illustrated in Fig. 6(b) finds the longest closed contour in the cable segmentation mask, removes its points lying on the image boundary and finds the cable backbone curve by interpolating the two remaining parallel contour lines. Fitting a spline curve to the backbone points estimates the normal vectors for computing the normal flow. Linearly interpolating the displacement of the two markers closest to a backbone point yields its motion. The remaining pixels of the cable segment obtain their flow estimate from their nearest backbone point. See Fig. 7 for a sample visualization of the ground truth optical and normal flow magnitude during a poking action.

d) Compositing and data augmentation: We composite each final clip from a static background image, a moving cable clip and one or more static clips or still images extracted from moving cable clips. We keep both the moving and static poking sticks in the compositons. One can generate a semi-three-dimensional scene of cables stacked on top of each other this way, see Fig. 8(a).

We manually downloaded CC0-licensed (public domain) background images from the internet. The search was biased towards textures, bushes or woods, and distractors (queries: texture, colorful texture, fractal texture, bushes, ropes, wires, pile). We divided the images into two classes: clutter and distractors. Distractors may be confused with hoses, cables, wires, or ropes. Clutter is everything else. See Fig. 8.

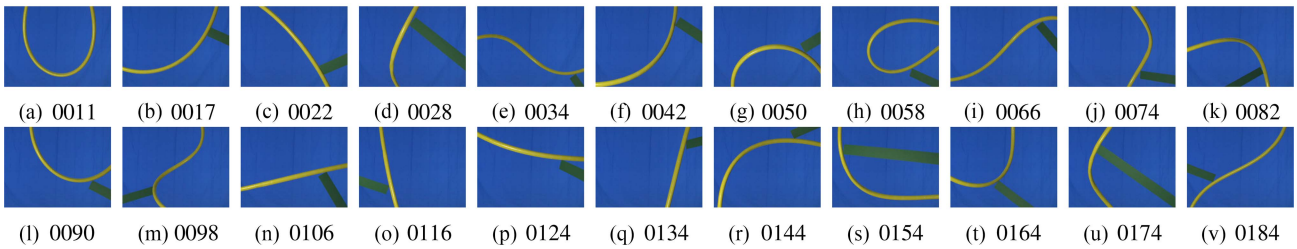


Fig. 9. The 22 cable configurations of the recorded raw dataset.

TABLE I
THE MAIN FEATURES OF THE COMPOSED DATASET

Property	Count	Comment
Cable configurations	22	
Cable densities	2	low (4-5 cables), high (10-11 cables)
Motion classes	4	poking, push/pull, lateral, static
Background classes	4	clutter, distractor, plain original, plain transformed color

Even though the backgrounds are often artificial textures or high-quality photographs, we wanted to reduce any JPEG artifacts and remaining sensor noise. Thus we downscaled each background image at least by a factor of two and extracted the center crop 640×480 pixels in size.

Foreground augmentation randomly alters the color of moving and static cables. It can transform hue, contrast, saturation and brightness; invert RGB colors, shuffle RGB channels, or convert to grayscale.

A sensor noise model adds artificial noise to the static background and still cable images to ensure that all image regions exhibit similar noise distributions. If we did not add noise, methods based on temporal image differencing could "segment" the moving cable by assuming that only the moving cable pixels were affected by variable sensor noise.

e) Sensor noise model: We use sRGBNoise [33], a model originally trained on images taken by five different smartphones [34]. The model generates noise conditioned on a noise-free image, the camera name, and ISO value. We collected a training set with the Basler camera to train its noise model. We treated the (downscaled) background images as the noise-free input to sRGBNoise at inference time. However, the real sensor noise already corrupted the still cable images. Therefore we applied a bilateral filter to suppress the noise before feeding them to sRGBNoise.

C. The Composed Dataset

We composed the final dataset from the 177 recorded clips (106 200 white-lit images in total). Each recorded clip shows a cable of a single configuration (i.e., a characteristic global shape), see Fig. 9, and a single motion class.

a) Dataset features: Table I summarizes the main features of the composed dataset. The motion classes are: poking the cable, pushing/pulling an endpoint, endpoint lateral motion, or static (no motion). The cable density relates to the number of cables overlaid in a composition.

TABLE II
THE DIVISION OF THE RECORDED CLIPS BY MOTION CLASS INTO THE THREE SPLITS (TRAINING, VALIDATION, TEST)

Motion class	Recorded	Training	Validation	Test
Poking	104	67	11	26
Push-pull	36	12	8	16
Lateral	16	3	3	10
Static	21	0	0	21

TABLE III
THE SIZE OF THE COMPOSED DATASET AND ITS SPLITS

	Train	Validation	Test	All
Clips	164	44	104	312
Images	98 399	26 407	62 381	187 187

b) Dataset splits: We composed the training, validation, and test dataset splits as follows. First, we divided the recorded clips into three mutually exclusive sets, one for each dataset split. The division satisfies the following constraints: (a) The images of each recorded clip are used in only one split. (b) In each split, each cable configuration is represented by at least one moving cable clip. (c) The number of recorded clips of each motion class in each split is specified in Table II. (d) The cable density classes are represented equally.

We used every recorded moving cable clip to create exactly two composed clips, each with a unique background and a unique combination of cable configurations. In a subset of the compositions, we also randomly transformed the colors of the cables or the plain background.

Table III presents the numbers of images and video clips in each composed dataset split. Each video clip is ten seconds long and consists of ca. 600 white-lit images.

IV. PROBABILISTIC MASKFLOWNET MOTION SEGMENTATION METHOD

Given a sequence of color images, poking stick segmentation masks, and a motion threshold τ, a motion segmentation algorithm detects cable motion with respect to the first (reference) image I_1 of the clip. The algorithm outputs a motion mask P_m for each image. The pixels p of cable segments in image I_j shifted by more than τ pixels away from their position in the reference image I_1 should be marked as moving in the motion mask, $P_m(p) = 1$. Poking stick pixels and all other pixels p should be marked as static, $P_m(p) = 0$.

We compare five cable motion segmentation methods. The first four of them are baseline methods based on off-the-shelf optical flow predictors, namely MaskFlownet [27], GMFlow [35], FlowFormer++ [36] and the OpenCV implementation of Farnebäck's optical flow algorithm [37]. To compute the motion segmentation masks, we added optical flow magnitude (L2-norm) thresholding to these methods.

The fifth method is our novel proposed method, MfnProb. To create MfnProb, we added probabilistic outputs [28] and thresholding to the MaskFlownet deep neural network architecture. Given a pair of noisy input images and trained (certain) network weights, MfnProb predicts noisy optical flow vectors. The probability distribution of a predicted optical flow vector is assumed to be multivariate Laplacian parametrized by location $\boldsymbol{\mu}$ and a diagonal covariance matrix Σ, $\sigma^2 = \operatorname{diag}\Sigma$. The network learns to predict the mean $\phi_p = \mu$ and the standard deviation $\sigma_p \in \mathbb{R}^2$ of each optical flow vector probability distribution given a pair of images.

The predicted standard deviation (or variance) has to be non-negative. To ensure that, Gast and Roth [28] proposed to predict the variance in log space, i.e., $\sigma^2 = \exp(\hat{\sigma}^2)$, where $\hat{\sigma}^2$ was the (log-space) output of the neural network. When we tried to train MfnProb with the exponential, the training diverged. Therefore we replaced the exponential with a softplus function, i.e. $\sigma = \ln(1 + \exp(\hat{\sigma}))$ if $\hat{\sigma} \leq 20$ and $\sigma = \hat{\sigma}$ otherwise, to ensure non-negative standard deviations.

The training loss function of a predicted optical flow vector $\phi_p \in \mathbb{R}^2$ given its ground truth $\phi_{gt} \in \mathbb{R}^2$ is

$$\left(\sum_{i=1}^{2} \frac{(\phi_{p,i} - \phi_{gt,i})^2}{\sigma_{p,i}^2} + \epsilon\right)^{0.5} + \sum_{i=1}^{2} \log(\sigma_{p,i}^2), \qquad (1)$$

which is proportional to the negative log-likelihood of the multivariate Laplacian distribution. We set

$$\sigma_{p,i} = \sigma_{\min} + \operatorname{softplus}(\hat{\sigma_{p,i}}). \qquad (2)$$

The index i runs over the two flow coordinates, horizontal and vertical. $\sigma_{p,i}$ is the standard deviation predicted by the network for the flow coordinate $\phi_{p,i}$. We set $\epsilon = 10^{-8}$ and $\sigma_{\min} = 10^{-2}$ to stabilize the training. We trained with the same training schedule on the same optical flow datasets as [27], namely FlyingChairs [38], FlyingThings3D [39], Sintel [40], KITTI [41], HD1K [42], [43].

In addition to thresholding the optical flow magnitude, MfnProb can utilize the predicted uncertainty to reduce false positives. The segmentation labels any pixel with uncertainty magnitude $\|\sigma_p\|_2 > \sigma_t$ as static. We empirically set the uncertainty threshold σ_t on the validation set to maximize the mean segmentation intersection over union (IoU). In practice, we argue it is safer to predict a static scene when too uncertain because reliable robot's actions, such as grasping, depend on precise true positive segmentation. When a segmentation algorithm has high precision but low recall, the robot can compensate for the low recall by trying multiple different motions until the segmentation succeeds. On the other hand, compensating for low precision is harder.

TABLE IV
MEAN EVALUATION METRICS ON THE TEST SET

Method	Recall	Precision	IoU	EPE (pixels)
MaskFlownet [27]	0.6098	0.6415	0.4079	1.22
MfnProb (ours)	<u>0.8768</u>	0.7324	<u>0.6560</u>	0.65
Farnebäck [37]	**0.8977**	0.3520	0.3335	1.56
GMFlow [35]	0.7202	<u>0.8077</u>	0.5925	<u>0.45</u>
FlowFormer++ [36]	0.7934	**0.8675**	**0.6932**	**0.44**
MaskFlownet FT (ours)	0.8286	0.8295	0.7022	0.37
MfnProb FT (ours)	**0.8762**	**0.8638**	**0.7673**	**0.32**

IoU – intersection over union, EPE – endpoint error of optical flow.
The bold values indicate the best performance and the the underlined values indicate the second best performance.

V. ALGORITHM EVALUATION PROCESS

Given a τ value, a predicted motion mask P_m, and the ground truth optical flow, the evaluation process computes standard segmentation quality metrics, namely the mean intersection over union (IoU), precision, and recall. Our experiments show that increasing the τ threshold above 10 pixels (up to 20) leads to significant decreases in both IoU and recall on the validation set of our dataset. On the other hand, the maximum noise level of the marker detector is 0.528 pixels for static markers. Therefore the evaluation varies τ from 1 to 20 pixels on the validation set and chooses the optimal τ^* value yielding the highest validation IoU. The evaluation reports the test set results given τ^*. In practice, a robot should try to move a cable as little as possible to preserve the cable topology and avoid hitting other cables by accident.

The evaluation also reports the mean endpoint error of the predicted optical flow (EPE) in pixels.

VI. EVALUATION RESULTS

Table IV shows the evaluation results of the cable motion segmentation methods on the test set of our dataset. Methods MaskFlownet FT and MfnProb FT are MaskFlownet and MfnProb fine-tuned on a mixture of Sintel, KITTI, HD1K, and the MovingCables training set. We evaluated the methods on the normal flow ground truth as the hoses in the clips have almost no texture, see Fig. 8(a). The optimal motion threshold τ values on the validation set were 2.5 pixels for MaskFlownet, 2.0 pixels for MfnProb, 1.0 pixel for Farnebäck, 1.0 pixel for GMFlow, and 1.5 pixels for FlowFormer++. The optimal uncertainty threshold of MfnProb was positive infinity, i.e., no high-uncertainty predictions had to be suppressed to maximize the validation IoU.

MfnProb outperforms MaskFlownet in all the evaluation metrics. The probabilistic training scheme reduced the overall mean EPE by almost half. Mean segmentation recall has improved by 68%, precision by 25%, and IoU by 42% simultaneously. MfnProb outperforms GMFlow in terms of IoU and recall but not precision. FlowFormer++ reaches the highest IoU among the methods not fine-tuned on MovingCables. MfnProb FT achieves the highest IoU overall. Sample segmentations are in Fig. 10. Our additional qualitative experiments on real-world videos without chroma keying or compositing indicate that all the motion segmentation methods generalize well to different cable textures (hoses, ropes, cables) and real backgrounds.

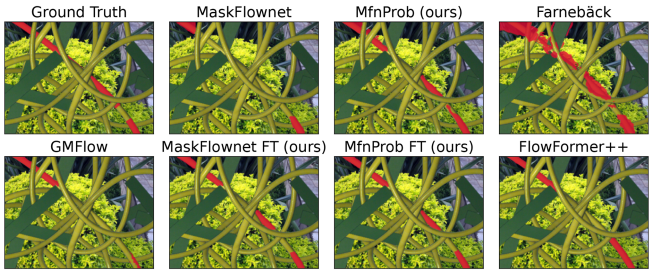


Fig. 10. Sample motion segmentation. Estimated and ground truth moving segments (at $\tau = 2.5$ pixels) are red.

TABLE V
MEAN WALL AND PROCESS RUNTIMES REQUIRED TO COMPUTE OPTICAL FLOW FOR A PAIR OF RGB VGA (640×480 PIXELS) IMAGES

Method	GPU wall (seconds)	CPU wall (seconds)	CPU process (seconds)
MaskFlownet [27]	**0.039**	2.6	8.6
MfnProb (ours)	0.041	2.6	8.7
Farnebäck [37]	N/A	**0.056**	**0.18**
GMFlow [35]	0.045	1.09	8.0
FlowFormer++ [36]	0.232	N/A	N/A

The CPU times apply only to algorithms running exclusively on the CPU. CPU process time is equal to user time plus system time or to the CPU wall time multiplied by the mean number of parallel threads used.
The bold values indicate the best performance and the the underlined values indicate the second best performance.

TABLE VI
MEAN IOUS ON THE TEST SET SEPARATELY FOR THREE BACKGROUND TYPES, CLUTTER, DISTRACTOR, AND PLAIN

Method	Clutter	Distractor	Plain
MaskFlownet [27]	0.4535	0.4739	0.2562
MfnProb (ours)	0.6889	0.7161	0.5322
Farnebäck [37]	0.3503	0.3060	0.3343
GMFlow [35]	0.5942	0.5889	0.5927
FlowFormer++ [36]	**0.7072**	**0.7396**	**0.6199**
MaskFlownet FT (ours)	0.7367	0.7525	0.5855
MfnProb FT (ours)	**0.7803**	**0.7942**	**0.7149**

The bold values indicate the best performance and the the underlined values indicate the second best performance.

Table V presents the runtime of each algorithm with batch size one. We obtained these results on a desktop computer with Intel Core i9-9900 K CPU (3.60 GHz) and NVIDIA GeForce RTX 2080 Ti GPU. The original and the probabilistic MaskFlownet networks are similarly computationally intensive, achieving runtimes around 0.040 s per image pair on the GPU and 2.6 s on the CPU. The CPU process times suggest that both networks demand approximately $48\times$ more CPU computation than Farnebäck's algorithm. FlowFormer++ is less suitable for real-time interactive perception than MfnProb as it is $5.7\times$ slower on a GPU.

We further evaluated the methods separately on clips with different background classes (clutter, distractor, plain), see Table VI. Clutter and distractor backgrounds yield comparably accurate segmentations. Plain backgrounds, however, tend to cause significantly more false positive segmentations by the neural networks in static areas, resulting in lower mean IoU. Replacing poorly textured plain backgrounds with texture-free solid colors completely confuses the neural networks, see Table VII. They falsely predict motion in almost the entire image. Fine-tuning MaskFlownet or MfnProb on MovingCables brings negligible improvements. By contrast, plain backgrounds do not affect Farnebäck's performance significantly. We think that the neural networks do not regularize towards the smallest flow at a pixel where many flow vectors have very similar matching costs.

TABLE VII
STATISTICS OF PER-CLIP MEAN IOUS ON 20 CLIPS WITH VARIOUS SOLID BACKGROUND COLORS

Method	Min.	Median	Mean	Max.
MaskFlownet [27]	0.0378	0.0433	0.0451	0.0531
MfnProb (ours)	0.0379	0.0443	0.0455	0.0530
Farnebäck [37]	**0.3977**	**0.4545**	**0.4508**	**0.4900**
GMFlow [35]	0.1033	0.1189	0.1193	0.1397
FlowFormer++ [36]	0.0573	0.0645	0.0667	0.0837
MaskFlownet FT (ours)	0.0567	0.0629	0.0631	0.0689
MfnProb FT (ours)	0.0526	0.0575	0.0605	0.0981

We kept one low density cable composition fixed for all the clips.
The bold values indicate the best performance and the the underlined values indicate the second best performance.

VII. DISCUSSION AND CONCLUSIONS

We have proposed a method to automatically annotate a real-world moving cable segmentation dataset with optical flow and segmentation masks thanks to UV fluorescent markers, controlled lighting, and chroma keying. Using the method, we have created the MovingCables dataset consisting of 312 video clips. The clips differ in their backgrounds, cable colors, numbers of overlaid cables, motion interaction types, or distinct combinations of cable configurations.

As an alternative to a real-world dataset, one could build a synthetic dataset in a simulator. For example, the Blender software can simulate chain-like rope dynamics.[2] It would likely require less manual work as one would not need to design and build any hardware setup. A simulator could simulate a cable in many different positions, such as lying on a desk or hanging freely. However, the cable appearance and the scene lighting would be synthetic. Furthermore, simulating realistic hose or cable dynamics may be more challenging than simulating a chain-like rope. Nevertheless, a synthetic moving cable dataset could complement the real-world dataset presented in this letter.

We have tested MaskFlownet, GMFLow, and FlowFormer++ off-the-shelf optical flow neural networks on our dataset and found that they can segment moving cables from a static background. We added uncertainty outputs to the MaskFlownet architecture and retrained it with a probabilistic loss function on standard optical flow datasets. This retrained MfnProb network has significantly improved the cable motion segmentation performance over MaskFlownet on our dataset. Fine-tuning MaskFlownet and MfnProb on MovingCables further improved

[2]https://blender.stackexchange.com/questions/97749/how-to-simulate-a-rope

the accuracy. Nevertheless, we believe that optical flow estimators should work reliably on any realistic visual input without fine-tuning.

Limitations: We have found that all the neural networks struggle with texture-free backgrounds. Furthermore, manipulating a cable in a cluttered environment can perturb neighboring cables, causing multiple moving cables. As our methods segment motion by thresholding the flow magnitude, they segment multiple moving cables as a single cable. We will address this limitation in future work.

ACKNOWLEDGMENT

The authors thank Libor Wagner, Jiří Sedlář, Karla Štěpánová, and several other colleagues from their lab.

REFERENCES

[1] A. Choi, D. Tong, B. Park, D. Terzopoulos, J. Joo, and M. K. Jawed, "mBEST: Realtime deformable linear object detection through minimal bending energy skeleton pixel traversals," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4863–4870, Aug. 2023.

[2] A. Caporali, K. Galassi, B. L. Zagar, R. Zanella, G. Palli, and A. C. Knoll, "RT-DLO: Real-time deformable linear objects instance segmentation," *IEEE Trans. Ind. Inform.*, vol. 19, no. 11, pp. 11333–11342, Nov. 2023.

[3] A. Caporali, K. Galassi, R. Zanella, and G. Palli, "FASTDLO: Fast deformable linear objects instance segmentation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9075–9082, Oct. 2022.

[4] T. Hodaň et al., "BOP challenge 2023 on detection, segmentation and pose estimation of seen and unseen rigid objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2024, pp. 5610–5619.

[5] R. Zanella, A. Caporali, K. Tadaka, D. D. Gregorio, and G. Palli, "Auto-generated wires dataset for semantic segmentation with domain-independence," in *Proc. IEEE Int. Conf. Comput. Control Robot.*, 2021, pp. 292–298.

[6] A. Caporali, R. Zanella, D. D. Greogrio, and G. Palli, "Ariadne+: Deep learning–based augmented framework for the instance segmentation of wires," *IEEE Trans. Ind. Inform.*, vol. 18, no. 12, pp. 8607–8617, Dec. 2022.

[7] A. Caporali et al., "A weakly supervised semi-automatic image labeling approach for deformable linear objects," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1013–1020, Feb. 2023.

[8] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2372–2379, Apr. 2020.

[9] M. Wnuk et al., "Kinematic multibody model generation of deformable linear objects from point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 9545–9552.

[10] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier, and A. Cherubini, "Robotic manipulation planning for shaping deformable linear objects with environmental contacts," *IEEE Robot. Automat. Lett.*, vol. 5, no. 1, pp. 16–23, Jan. 2020.

[11] J. Zhu et al., "Vision-based manipulation of deformable and rigid objects using subspace projections of 2D contours," *Robot. Auton. Syst.*, vol. 142, Aug. 2021, Art. no. 103798.

[12] A. Keipour, M. Bandari, and S. Schaal, "Deformable one-dimensional object detection for routing and manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4329–4336, Apr. 2022.

[13] V. Viswanath et al., "Disentangling dense multi-cable knots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3731–3738.

[14] K. Shivakumar et al., "SGTM 2.0: Autonomously untangling long cables using interactive perception," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5837–5843.

[15] K. Lv, M. Yu, Y. Pu, X. Jiang, G. Huang, and X. Li, "Learning to estimate 3-D states of deformable linear objects from single-frame occluded point clouds," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7119–7125.

[16] S. Zhaole, H. Zhou, L. Nanbo, L. Chen, J. Zhu, and R. B. Fisher, "A robust deformable linear object perception pipeline in 3D: From segmentation to reconstruction," *IEEE Robot. Automat. Lett.*, vol. 9, no. 1, pp. 843–850, Jan. 2024.

[17] P. Sundaresan et al., "Untangling dense non-planar knots by learning manipulation features and recovery policies," in *Proc. 17th Robot.: Sci. Syst.*, 2021. [Online]. Available: https://www.roboticsproceedings.org/rss17/p013.html.

[18] X. Zhang, Y. Domae, W. Wan, and K. Harada, "Learning efficient policies for picking entangled wire harnesses: An approach to industrial bin picking," *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 73–80, Jan. 2023.

[19] J. Bohg et al., "Interactive perception: Leveraging action in perception and perception in action," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1273–1291, Dec. 2017.

[20] C. J. Tsikos and R. K. Bajcsy, "Segmentation via manipulation," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 306–319, Jun. 1991.

[21] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1377–1382.

[22] C. D. Singh, N. J. Sanket, C. M. Parameshwara, C. Fermüller, and Y. Aloimonos, "NudgeSeg: Zero-shot object segmentation by repeated physical interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2714–2712.

[23] W. Boerdijk et al., "Self-supervised object-in-gripper segmentation from robotic motions," in *Proc. 4th Conf. Robot Learn.*, 2020, pp. 1231–1245.

[24] T. Patten, M. Zillich, and M. Vincze, "Action selection for interactive object segmentation in clutter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 6297–6304.

[25] A. Eitel, N. Hauff, and W. Burgard, "Self-supervised transfer learning for instance segmentation through physical interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4020–4026.

[26] A. Price, K. Huang, and D. Berenson, "Fusing RGBD tracking and segmentation tree sampling for multi-hypothesis volumetric segmentation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9572–9578.

[27] S. Zhao et al., "MaskFlownet: Asymmetric feature matching with learnable occlusion mask," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6278–6287.

[28] J. Gast and S. Roth, "Lightweight probabilistic deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3369–3378.

[29] S. Baker et al., "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, Nov. 2010.

[30] K. Takahashi and K. Yonekura, "Invisible Marker: Automatic annotation of segmentation masks for object manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8431–8438.

[31] B. Thananjeyan, J. Kerr, H. Huang, J. E. Gonzalez, and K. Goldberg, "All you need is LUV: Unsupervised collection of labeled images using UV-fluorescent markings," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 3241–3248.

[32] R. Szeliski, *Computer Vision: Algorithms and Applications*. London, U.K.: Springer, 2011.

[33] S. Kousha et al., "Modeling sRGB camera noise with normalizing flows," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 17463–17471.

[34] A. Abdelhamed et al., "A high-quality denoising dataset for smartphone cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1692–1700.

[35] H. Xu et al., "Unifying flow, stereo and depth estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13941–13958, Nov. 2023.

[36] X. Shi et al., "FlowFormer++ : Masked cost volume autoencoding for pretraining optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 1599–1610.

[37] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. Image Anal.*, 2003, pp. 363–370.

[38] A. Dosovitskiy et al., "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.

[39] N. Mayer et al., "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4040–4048.

[40] D. J. Butler et al., "A naturalistic open source movie for optical flow evaluation," in *Proc. Comput. Vis. ECCV: 12th Eur. Conf. Comput. Vis.*, 2012, pp. 611–625.

[41] A. Geiger et al., "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Aug. 2013.

[42] D. Kondermann et al., "Stereo ground truth with error bars," in *Proc. Comput. Vis.–ACCV : 12th Asian Conf. Comput. Vis.* 2015, pp. 595–610.

[43] D. Kondermann et al., "The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2016, pp. 19–28.