# Feature Learning from Image Markers for Object Delineation

Italos Estilon de Souza*, Barbara C. Benato† and Alexandre Xavier Falcão‡
Institute of Computing, University of Campinas
Campinas, Brazil
Email: *italos.souza@ic.unicamp.br, †barbara.benato@ic.unicamp.br, ‡afalcao@ic.unicamp.br

***Abstract*—Convolutional neural networks (CNNs) have been used in several computer vision applications. However, most well-succeeded models are usually pre-trained on large labeled datasets. The adaptation of such models to new applications (or datasets) with no label information might be an issue, calling for the construction of a suitable model from scratch. In this paper, we introduce an interactive method to estimate CNN filters from image markers with no need for backpropagation and pre-trained models. The method, named FLIM (*feature learning from image markers*), exploits the user knowledge about image regions that discriminate objects for marker selection. For a given CNN's architecture and user-drawn markers in an input image, FLIM can estimate the CNN filters by clustering marker pixels in a layer-by-layer fashion – i.e., the filters of a current layer are estimated from the output of the previous one. We demonstrate the advantages of FLIM for object delineation over alternatives based on a state-of-the-art pre-trained model and the Lab color space. The results indicate the potential of the method towards the construction of explainable CNN models.**

## I. Introduction

Deep learning has enabled important advances in different areas of knowledge. In particular, Convolutional Neural Networks (CNNs) have revolutionized computer vision, with applications to various tasks, such as object detection and identification. Since the first models called attention with competitive results (e.g., AlexNet in the ImageNet competition), CNNs have become more complex, with a higher number of layers and a considerably higher number of parameters for estimation. These models are considered as "black-boxes", implying that one cannot explain their decisions. Explainable artificial intelligence (XAI) has appeared to address the problem and avoid wrong interpretations of the results [1]–[4].

In methods of XAI, however, the importance of human participation during the model's training process has called little attention yet. A network designer must choose a suitable architecture and the training hyperparameters with limited control over the quality of the model's estimated weights. The designer might also have to count on an expert in the application domain (user), who must provide a dataset with a sufficient number of correctly annotated samples for the model's training. Ideally, the user and designer should actively participate in the data annotation and training processes, both assisted by the machine, to increase human understanding and control, reduce effort, and improve interpretation of the results.

In this paper, we fill an essential part of the gap above by presenting an interactive method where the user (also a designer) draws strokes (markers) in parts that represent object and background in an image, and the filters of a CNN are learned from those markers with no need for backpropagation. We call this approach *feature learning from image markers* (FLIM) and demonstrate its potential for object delineation. The method extracts a patch around each marker pixel and finds clusters of patches per marker to derive the filters. The center of each group defines one filter that will enhance and extract features from the selected region. The network is trained forward in a layer-by-layer fashion. The filters of each convolutional layer are obtained from the previous layer's output by the same procedure – clustering patches of marker pixels. The user only needs to put markers in the input image, and, for segmentation, image resolution remains the same along with the layers to preserve boundary information.

Note that our goal here is not to propose a new solution for interactive segmentation. We aim to explain how feature learning can be effectively solved from a simple and intuitive user-interaction procedure and demonstrate the FLIM's potential to develop and improve image segmentation methods. For that, we use a sequence of convolutional layers for feature extraction and solve object delineation by a watershed transform from markers [5] on a gradient image derived from FLIM features. By fixing the object delineation method, we then compare different feature extraction procedures.

We have recently demonstrated FLIM for image classification [6], but the work differs from the current one in several aspects besides the application – coconut-tree image classification. In that work, the user put markers in a few training images from each class, the filters are derived by clustering patches per marker, but a single network is generated for all images. The network has a single convolutional layer with max-pooling and strides greater than one, followed by batch normalization and image classification by a multilayer perceptron. In the current work, a suitable feature extractor with one or two convolutional layers is computed per image based on user-selected markers, and a watershed transform completes image segmentation.

The remainder of the text is divided as follows. In Section II, we review works related to user-interaction for image segmentation. In Section III, we introduce basic definitions and present FLIM. Experimental results that demonstrate FLIM can be more effective than baselines are presented in Section IV. Finally, conclusion and future work are described

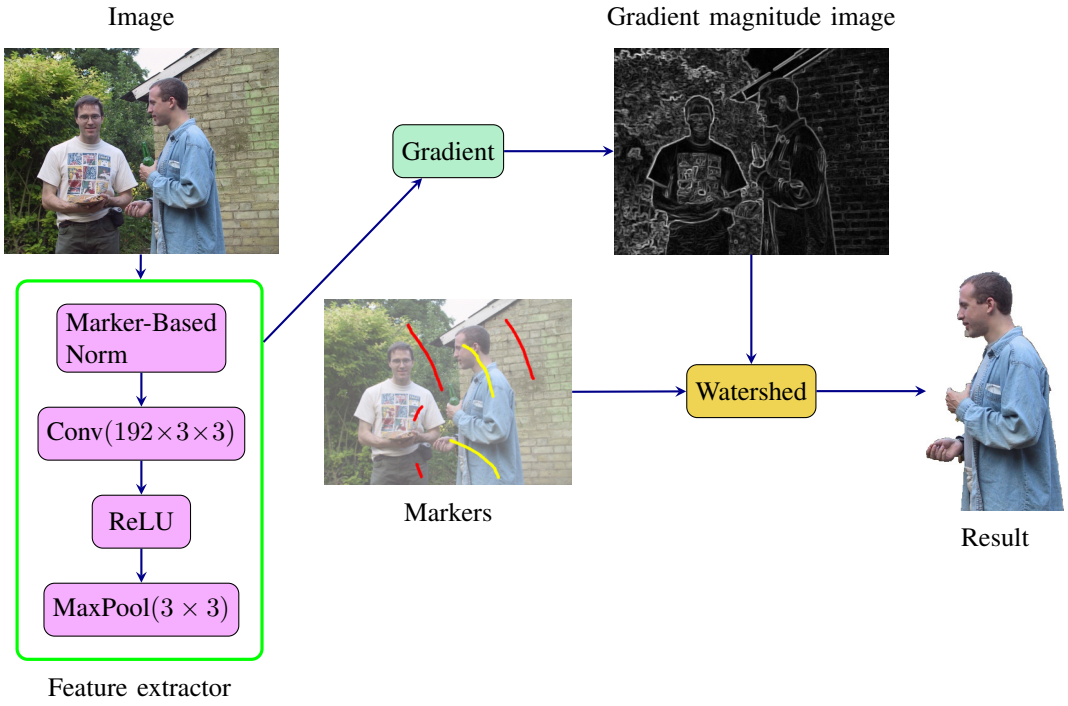



Fig. 1. For a given image, set of user-drawn markers, and desired number $m' = 192$ of filters, FLIM learns the weight of the filters (each with 3 channels and $3 \times 3$ pixels), a gradient magnitude image is estimated from the output of the layer, and a watershed transform delineates the object from the same markers on the gradient magnitude image. The user may add convolutional layers and markers to improve gradient and so image segmentation.

in Section V.

## II. Related works

Methods for image segmentation may be divided into automatic [7]–[11] and interactive [12]–[17] approaches. While interactive methods aim to reduce the user effort and the total time spent by the user to complete segmentation, automatic methods aim to learn models for object localization and delineation from an annotated set of training images. In both cases, human interaction is required. However, interactive segmentation approaches should also explore human knowledge during the training process to improve the model. In [18], the authors address the problem of object enhancement from image markers during interactive segmentation. The method does not require a pre-annotated dataset and selects marker pixels for object enhancement by a pattern classifier. However, there is no feature learning model, and the performance of the classifier strongly depends on the choice of suitable marker pixels. On the other hand, the idea is important to investigate the impact of marker pixels on feature learning in future work.

Since [19], most works on semantic segmentation are based on encoder-decoder models, also known as fully convolutional neural (FCN) networks [8], [9], [11]. The encoder is responsible for local image feature extraction and aggregation, while the decoder combines the features into a higher-level image content representation. Despite their popularity, such methods usually require a large set of training images pre-annotated by humans – a quite costly task. In interactive methods, a pre-trained model on some large annotated dataset can start the segmentation process, and the user may add markers to correct segmentation and improve the model [16].

To reduce the amount of annotated data required to train deep learning models for segmentation, Yang et al. [20] proposed an active learning framework based on pre-trained FCNs for medical image segmentation. The method suggests samples for user annotation. In [21], the selection of samples for user annotation is decided by reinforcement learning using the maximization of intersection-over-union as an optimization criterion.

Other works have actively pursued deep learning for interactive segmentation [12], [13], [15], [16]. Such methods have in common the use of an FCN model to estimate an object probability map and solve segmentation by thresholding that map. Xu et al. [13] introduce the idea of extending the input with Euclidean distance maps of user's clicks to improve the pre-trained FCN model and so object segmentation from the user's corrections. Jang et al. [15] avoided updating the pre-trained model for each user click so as not to lose the knowledge learned in the training phase – only the interaction maps are updated. During segmentation, multiple forward- and backward-feed iterations may be required. To reduce the computational cost of such operations, Sofiiuk et al. [16] propose a variant that avoids backward-feed iterations through the entire network.

The above methods explore user corrections to improve segmentation, but the resulting FCN models are still not

explainable. Such methods are also not feasible whenever annotated datasets are scarce, and pre-trained models are not suitable. Our approach for feature learning from image markers aims to improve the explanation of the resulting models and cope with the possible absence of previously annotated datasets.

## III. FLIM: Feature Learning from Image Markers

In convolutional neural networks (CNNs), feature extraction is performed by a sequence of convolutional layers. In addition to the convolution with a filter bank followed by activation, each layer may also have other operations (e.g., pooling and batch normalization). The filters in these layers are meant to highlight image regions relevant for better class separation. Fully-connected layers of a multilayer perceptron are then used to reduce the feature space dimension and decide each input image's class. CNN architectures, famous for accurate image classification, present several layers and thousands of filters, leading to the problem of requiring large annotated training sets for suitable weight optimization by backpropagation. We explore the human knowledge to indicate image regions suitable for filter estimation in CNNs (without pre-annotated dataset, pre-trained model, and backpropagation) and use the extracted image features for object delineation. The method is a forward-feed training approach, called FLIM (*Feature Learning from Image Markers*).

Let $\mathcal{D}$ be a set of images $I$ with $m$ channels each for object delineation. Let $\phi$ be a CNN architecture with only convolutional layers to extract image features from $I$. Each layer $l$ of $\phi$ consists of *marker-based normalization*, convolution with a set $\mathcal{F}_l$ of filters, a ReLU activation, and a max-pooling operation with stride one to preserve boundary information (see example in Figure 1). Let $k \times k \times m$ be the shape of each filter $F \in \mathcal{F}_l$. Given a filter $F \in \mathcal{F}_l$, the convolution between $I$ and $F$ at a pixel $p$ is the inner product between $\mathrm{vec}\, P_I(p)$ and $\mathrm{vec}\, F$, where $P_I(p)$ is a $k \times k$ patch centered at $p$ with $m$ channels and vec is the vectorization operation. If $P_I(p)$ contains a pattern detected by $F$, then the convolution at $p$ must have a positive value, and its value is negative otherwise. That is, $\mathrm{vec}(F)$ is the normal vector of a hyperplane in $\mathbb{R}^{k \times k \times m}$, and patches with patterns detected by $F$ are represented by vectors on the positive side of that hyperplane. We want these filters to highlight image features from $I$ that are important to segment a given object. Therefore, the user places markers in regions of $I$ that discriminate object and background properties, and these markers are used to find the set $\mathcal{F}_l$. The weight estimation of the filters in $\mathcal{F}_l$ and marker-based normalization are explained next.

For $c$ objects of interest, a pixel $p$ of $I$ may be assigned to one label $\lambda(p) = i \in \{0, 1, 2, \cdots, c\}$ where $\lambda(p) = 0$ indicates background. Let $\mathcal{M}_I$ be the set of pixels for markers placed in image $I$, and $\mathcal{P}_i$ be the set of all patches around pixels of $\mathcal{M}_I$ that have label $i$.

$$\mathcal{P}_i = \bigcup_{I \in \mathcal{D}, p \in \mathcal{M}_I, \lambda(p) = i} P_I(p). \tag{1}$$

One can apply a clustering ($K$-means [22] in this work) operation on $\mathcal{P}_i$ to detect the patterns that best represent label $i$. One can also execute clustering per marker rather than per label to cope with the case of multiple markers per object, each covering a region with distinct properties. We use the latter option for binary segmentation (i.e., $\lambda(p) \in \{0, 1\}$). The centroid of each cluster defines the weights of a filter $F \in \mathcal{F}_l$, representing a pattern associated with some label $i$. The user specifies the number of clusters per label $i$ (or per marker, as mentioned above) to capture all relevant patterns for segmentation.

To mostly enhance the regions detected by a filter $F$ at its corresponding output channel, the image $I$ must have its pixel values normalized as follows. Let $\mathcal{P} = \bigcup_{i \in \{0,1,2,\ldots,c\}} \mathcal{P}_i$ be the set of all patches. Their mean and standard deviation (channel-wise) are calculated for centralization and standardization, by subtracting the mean values from the pixel values with the result divided by the standard deviation. We call this operation *marker-based normalization*. It must be included at the beginning of each convolutional layer to extract patches and estimate the filter weights by clustering. We also force $\| \mathrm{vec}(F) \| = 1$ to avoid preferences among filters.

Each image $I \in \mathcal{D}$ derives a CNN with one or more layers, as specified by the user, whose filters are estimated from user-drawn markers. To illustrate the advantages of this feature learning approach, we compute a gradient magnitude image and apply a watershed transform from the same markers to delineate the object, as shown in Figure 1. The input image $I$, as well as the output image $O$ from FLIM, are multi-channel images, and gradient magnitude images can be similarly computed from them. Let $m'$ be the number of filters in the last layer of the CNN, such that $O_k$, $k = 1, 2, \ldots, m'$, are the channels of $O$. Let $A(p)$ be the set of adjacent pixels $q = (x_q, y_q)$ of a pixel $p = (x_p, y_p)$ such that $\|q - p\| \leq r$ (e.g., $r = 1$). A gradient image $G_O(x, y)$ at a pixel $p$ is defined as

$$G_O(p) = \frac{1}{|A(p)|} \sum_{\forall q \in A(p)} \|O_k(q) - O_k(p)\|_2. \tag{2}$$

FLIM is applied to learn filters in a layer-by-layer fashion. Thus, the output of layer $l$ is used to learn the filters of layer $l + 1$ by clustering, as described above. Note that, once the user defines the CNN architecture, the whole process is fully automatic. However, the user may add new markers and select filters manually, based on the visualization of its output channel. For instance, the user may decide to simplify the CNN by eliminating redundant filters – i.e., filters that produce similar output channels. In Figure 2, we illustrate one example to which changes in the model's architecture improves the result with no additional markers. It also calls attention to the importance of activation and gradient visualization for better explaining the results.

## IV. Experimental Results

The watershed transform from the same set of user-drawn markers, and gradient image will present different segmen-

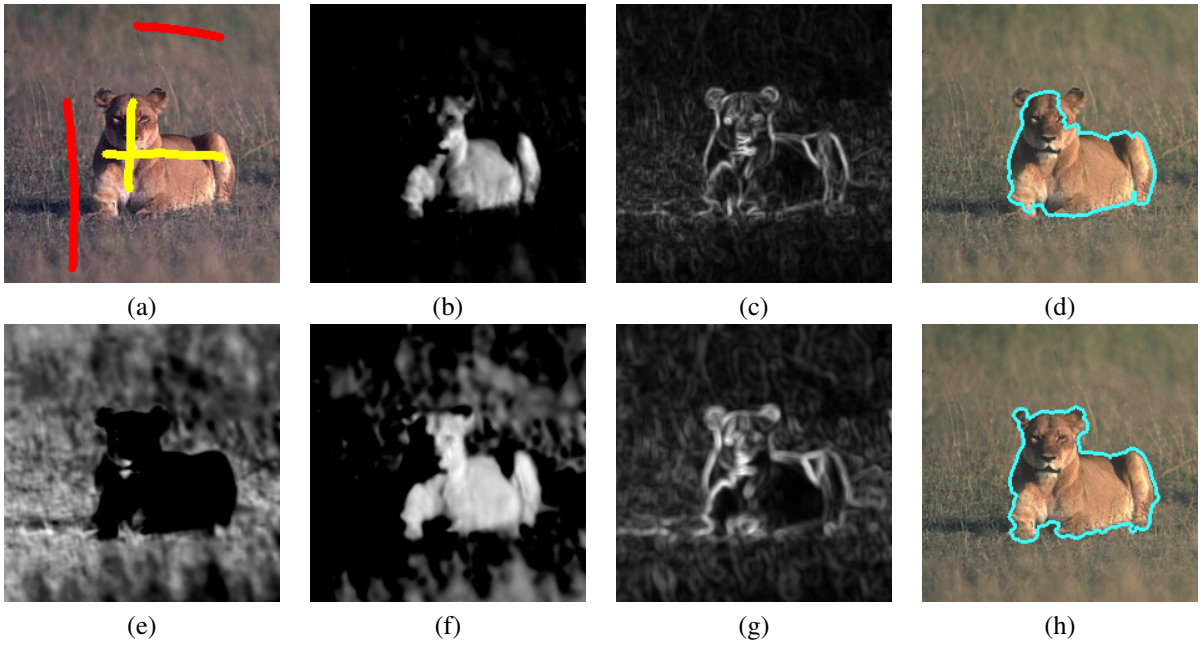



Fig. 2. (a) Input image with user-drawn markers. The user specifies the first layer with 8 filters of $3 \times 3$ pixels per marker but eliminates redundant filters whose inner product between them is above 0.85. (b) One example of an activation image for one filter of the first layer. (c) The gradient image from the output of 6 selected filters in the first layer and (d) the resulting watershed segmentation. The user specifies a second layer with 64 filters of $5 \times 5$ pixels per marker and applies the same threshold to eliminate redundant filters. (e,f) Examples of activation images from two filters of the second layer. (g) The gradient image from the output of 34 selected filters in the second layer and (h) the resulting watershed segmentation.

tation performances depending on the image features used for gradient computation in Equation 2. In this section, we evaluate the performance of the watershed transform when FLIM obtains the image features for gradient computation, image features are the Lab colors of the input image, or image features are obtained at one of two evaluated outputs of a network named DeeplabV3 [23] – well known as very effective for semantic segmentation [24].

The Geostar dataset [25] was chosen for the segmentation experiments. It consists of 151 color images of up to $800\times$ 600 pixels each, a set of user-drawn markers per image, and the ground-truth segmentation masks. Since the idea is to select markers in regions that discriminate object and background, we decided to create our own set of user-drawn markers. The number of markers and size varies from image to image, but they are all 5-pixel-wide. These markers are fixed as default in all methods. However, in order to make clear that our marker set is better than the original one, we present the results of FLIM using both marker sets. FLIM$_{mk1}$ uses the original marker set from Geostar, and FLIM$_{mk2}$ uses the default marker set. We evaluate a single-shot segmentation by the watershed transform using those markers and gradient images from each method. The segmentation results are evaluated by *Intersection Over Union* (IoU) between segmented image and ground-truth mask. As higher is the IoU value, better is the segmentation result.

We call INPUT the case of no feature learning – i.e., the gradient is obtained in the Lab colors of the input image. Since DeeplabV3 [23] uses an encoder to reduce image resolution (losing boundary information) and a decoder to obtain an object probability map (increasing object enhancement), we evaluate image features from the outputs of its first and last blocks (it contains several blocks, and a block may contain tens of convolutional layers). The output of the first block (DLAB$_{first}$) is the one with the best results in the encoder, while the output of the last block (DLAB$_{last}$) should be the one that best enhances the object's boundary in the gradient image. DeeplabV3 was also pre-trained in 11,530 images from the PASCAL dataset, and it is available in the torchvision package. For pre-processing, DeeplabV3 uses the images normalized by the mean and standard deviation of each channel.

Like any other network, FLIM still requires the user specification of the model's architecture. For a given set of user-drawn markers, FLIM may take some seconds in CPU to learn the filter weights of a network with a few convolutional layers and strides 1. Therefore, it should be feasible to develop strategies that optimize the model's architecture in a layer-by-layer fashion based on the pixel activations at object and background markers. Nevertheless, we decided to evaluate the segmentation pipeline for 90 different models per image by varying the number of layers, size of filters, and the number of filters per marker. The values of these hyperparameters are presented in Table I. Assuming that architecture optimization is feasible, we selected the best model per image in order to compare its result with the ones of using INPUT, DLAB$_{first}$, and DLAB$_{last}$. The results are presented in Table II. First, they show that FLIM$_{mk2}$ is better than FLIM$_{mk1}$, indicating

TABLE I
TESTED HYPERPARAMETERS FOR FLIM-BASED MODEL ARCHITECTURE.

| Hyperparameter | Possible values |
|---|---|
| Number of layers | 1, 2 |
| Kernel size | 3, 5, 7 |
| Number of kernels per marker | 8, 16, 32 |

TABLE II
MINIMUM, MAXIMUM, MEDIAN, MEAN, AND STANDARD DEVIATION OF IOU FOR THE SEGMENTATION OF ALL 151 IMAGES USING INPUT, $\text{DLAB}_{first}$, $\text{DLAB}_{last}$, $\text{FLIM}_{mk1}$, AND $\text{FLIM}_{mk2}$. THE BEST RESULTS PER METRIC ARE IN BOLD.

| Method | min | max | median | mean | std |
|---|---|---|---|---|---|
| INPUT | 0.084 | **1.000** | 0.601 | 0.590 | 0.216 |
| $\text{DLAB}_{first}$ | 0.134 | 0.977 | 0.583 | 0.585 | 0.205 |
| $\text{DLAB}_{last}$ | 0.040 | 0.824 | 0.403 | 0.395 | 0.179 |
| $\text{FLIM}_{mk2}$ | **0.406** | 0.998 | **0.796** | **0.784** | **0.147** |
| $\text{FLIM}_{mk1}$ | 0.220 | 0.999 | 0.746 | 0.713 | 0.194 |

that our marker set is more suitable than the one from Geostar. Second, for both marker sets, FLIM can provide more effective segmentation with very simple models than using the image features from DeepLabV3 and the Lab color space.

Table III presents the IoU values for each image in GeoStar when using INPUT, $\text{DLAB}_{first}$ (better than $\text{DLAB}_{last}$), and $\text{FLIM}_{mk2}$ (better than $\text{FLIM}_{mk1}$). One can observe that $\text{FLIM}_{mk2}$ can consistently achieve the best IoU for most images. When INPUT and $\text{DLAB}_{first}$ achieve higher IoU than $\text{FLIM}_{mk2}$, the difference is usually low.

### A. Implementation details for FLIM

Feature extractors were implemented using the Pytorch package [26]. For clustering, we use the implementation of K-means with mini-batches available in the Scikit-Learning package [27]. We set the maximum number of iterations to 100 and stopped after 10 iterations with no improvement. The batch size was 100, and the clustering was initialized with "k-means++". The remaining parameters were set with their default values. For the watershed transform, we use the implementation available in the Scikit-Image package [28] using 8-neighborhood for label propagation.

### B. Discussion

Tables II and III indicate that FLIM can consistently learn relevant filters for given network architecture, improving image features as compared to the Lab color space and feature spaces obtained by DeepLabV3, a pre-trained model. The results between $\text{DLAB}_{first}$ and $\text{DLAB}_{last}$ show that the decoder cannot cope with the loss of spatial resolution (boundary information) in the encoder, impairing object delineation. Note that, the U-shape of encoder-decoder models is usually adopted for semantic segmentation. They might improve object detection, but this raises a question about their effectiveness for delineation, and both are required for accurate segmentation. The results between $\text{FLIM}_{mk1}$ and $\text{FLIM}_{mk2}$ show that user-draw markers play a role in filter estimation, and the comparison between the FLIM-based models and the baselines indicates that a specific model per image can considerably improve effectiveness using very simple architectures.

Figure 2 shows that an interactive choice of the model's architecture can improve segmentation with no additional markers. When drawing strokes in parts that discriminate object and background, FLIM can estimate filters that will enhance selected parts of both (Figures 2b, 2e, and 2f). Irrelevant and redundant filters can be eliminated, simplifying the convolutional layer of the model under construction. As a consequence, the gradient improvement from layer 1 (Figure 2c) to 2 (Figure 2g) was enough to improve segmentation.

For the experiments, marker selection did not follow any particular strategy rather than drawing markers in parts that discriminate object and background. Some examples are presented in Figure 3. Note that external markers should be around the object. Markers, such as those selected on the sky, water, and reef in Figure 3k are irrelevant for that result. Markers near the boundary in parts where the object and background have similar properties are also important (e.g., Figure 3e). Such selection creates redundant filters, but it avoids errors by the watershed transform. Most errors in Figure 3 have been caused by not following a better strategy for marker selection.

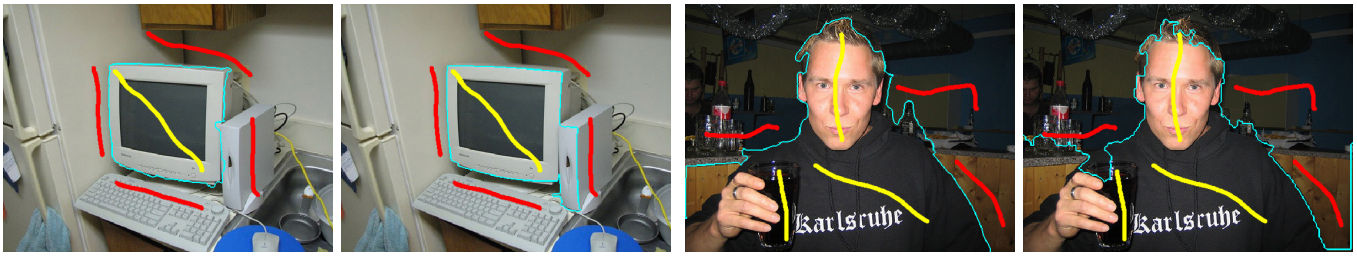
Figure 4 shows that a suitable strategy for marker selection combined with the incremental construction of FLIM-based models can be a practical methodology for interactive segmentation. Redundant filters are eliminated as in the example of Figure 2 from a set of 64 estimated filters per marker.

## V. CONCLUSION AND FUTURE WORK

We introduced an intuitive and interactive method, named FLIM, to estimate CNN filters from image markers. As far as we know, FLIM is the first feature learning approach for the construction of neural networks, which does not require pre-training from large annotated datasets and backpropagation. Its potential has been demonstrated for image classification [6] and, in this paper, for object delineation. In both applications, the results are promising such that future work involves several tasks.

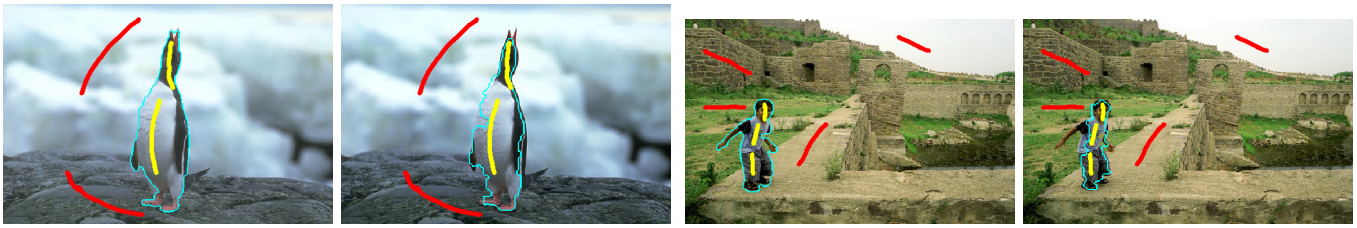
We must further explore clustering algorithms, develop techniques to identify and eliminate irrelevant and redundant filters, and investigate data visualization methods to facilitate marker selection and architecture choice while constructing the model in a layer-by-layer fashion. We intend to develop interactive image segmentation methods based on FLIM, verify the model's updates as the user selects new markers, and compare them with other interactive segmentation methods based on deep learning, such as f-BRS [16]. It seems that strategies to avoid irrelevant and redundant filters will be necessary at this point. Another interesting direction is to use a multilayer perceptron for pixel-wise classification from the features extracted with FLIM to combine the gradient of its object map with the gradient computed from the network features in the

(a) FLIM - IoU: 0.878. (b) DLab$_{first}$ - IoU: 0.868. (c) FLIM - IoU: 0.808. (d) INPUT - IoU: 0.815.



(e) FLIM - IoU: 0.929. (f) INPU - IoU: 0.808. (g) FLIM - IoU: 0.918. (h) DLab$_{first}$ - IoU: 0.867.

(i) FLIM - IoU: 0.946. (j) DLab$_{first}$ - IoU: 0.413. (k) FLIM - IoU: 0.945. (l) DLab$_{first}$ - IoU: 0.867.



(m) FLIM - IoU: 0.900. (n) DLab$_{first}$ - IoU: 0.763. (o) FLIM - IoU: 0.931. (p) DLab$_{first}$ - IoU: 0.645.

Fig. 3. Segmentation results using FLIM$_{mk2}$ and the best baseline for each image.

TABLE III
IoU VALUES PER IMAGE USING THE METHODS FLIM$_{mk2}$, DLAB$_{first}$, AND INPUT. THE BEST RESULTS PER IMAGE ARE IN BOLD.

| Image | FLIM$_{mk2}$ | DLab$_{first}$ | INPUT |
|---|---|---|---|
| 001 | **0.900** | 0.763 | 0.565 |
| 002 | **0.853** | 0.699 | 0.758 |
| 003 | **0.848** | 0.419 | 0.696 |
| 004 | **0.739** | 0.666 | 0.666 |
| 005 | 0.906 | 0.834 | **0.911** |
| 006 | **0.902** | 0.884 | 0.393 |
| 007 | **0.882** | 0.583 | 0.807 |
| 008 | **0.859** | 0.742 | 0.843 |
| 009 | 0.967 | 0.936 | **0.971** |
| 010 | 0.649 | **0.662** | 0.661 |
| 011 | 0.791 | 0.741 | **0.792** |
| 012 | **0.650** | 0.550 | 0.639 |
| 013 | **0.595** | 0.519 | 0.449 |
| 014 | **0.739** | 0.385 | 0.638 |
| 015 | **0.748** | 0.583 | 0.438 |
| 016 | **0.946** | 0.413 | 0.228 |
| 017 | **0.466** | 0.383 | 0.304 |
| 018 | **0.754** | 0.328 | 0.419 |
| 019 | **0.985** | 0.954 | 0.915 |
| 020 | 0.972 | 0.894 | **0.975** |
| 021 | **0.995** | 0.947 | 0.846 |
| 022 | **0.922** | 0.638 | 0.752 |
| 023 | **0.841** | 0.714 | 0.247 |
| 024 | **0.671** | 0.377 | 0.484 |
| 025 | 0.713 | **0.725** | 0.355 |
| 026 | 0.491 | 0.528 | **0.659** |
| 027 | **0.478** | 0.281 | 0.420 |
| 028 | **0.739** | 0.250 | 0.484 |
| 029 | **0.533** | 0.227 | 0.263 |
| 030 | **0.673** | 0.583 | 0.608 |
| 031 | 0.808 | 0.704 | **0.815** |
| 032 | **0.904** | 0.532 | 0.894 |
| 033 | **0.860** | 0.624 | 0.684 |
| 034 | **0.634** | 0.436 | 0.355 |
| 035 | **0.776** | 0.651 | 0.316 |
| 036 | **0.720** | 0.270 | 0.265 |
| 037 | **0.713** | 0.385 | 0.450 |
| 038 | **0.849** | 0.805 | 0.737 |
| 039 | **0.898** | 0.373 | 0.507 |
| 040 | **0.634** | 0.230 | 0.389 |
| 041 | **0.643** | 0.351 | 0.378 |
| 042 | **0.796** | 0.670 | 0.772 |
| 043 | **0.566** | 0.467 | 0.424 |
| 044 | 0.922 | 0.854 | **0.932** |
| 045 | 0.443 | **0.516** | 0.333 |
| 046 | **0.814** | 0.217 | 0.607 |
| 047 | **0.821** | 0.581 | 0.742 |
| 048 | **0.911** | 0.720 | 0.737 |
| 049 | **0.853** | 0.727 | 0.703 |
| 050 | **0.944** | 0.567 | 0.634 |
| 051 | **0.628** | 0.529 | 0.492 |
| 052 | **0.815** | 0.486 | 0.626 |
| 053 | **0.605** | 0.395 | 0.403 |
| 054 | **0.813** | 0.697 | 0.714 |
| 055 | 0.629 | **0.663** | 0.636 |
| 056 | 0.784 | **0.919** | 0.576 |
| 057 | **0.675** | 0.657 | 0.652 |
| 058 | **0.789** | 0.286 | 0.216 |
| 059 | **0.675** | 0.580 | 0.601 |
| 060 | **0.878** | 0.868 | 0.806 |
| 061 | **0.792** | 0.313 | 0.705 |
| 062 | 0.452 | **0.512** | 0.382 |
| 063 | **0.912** | 0.410 | 0.426 |
| 064 | **0.726** | 0.266 | 0.219 |
| 065 | **0.613** | 0.481 | 0.510 |
| 066 | 0.970 | 0.844 | **0.972** |
| 067 | **0.729** | 0.628 | 0.725 |
| 068 | **0.632** | 0.439 | 0.488 |
| 069 | **0.666** | 0.635 | 0.642 |
| 070 | **0.909** | 0.908 | 0.908 |
| 071 | **0.578** | 0.401 | 0.447 |
| 072 | **0.945** | 0.864 | 0.785 |
| 073 | **0.920** | 0.434 | 0.425 |
| 074 | 0.890 | 0.902 | **0.977** |
| 075 | 0.406 | **0.419** | 0.301 |
| 076 | 0.774 | 0.734 | **0.776** |
| 077 | **0.788** | 0.760 | 0.776 |
| 078 | **0.907** | 0.694 | 0.713 |
| 079 | **0.910** | 0.895 | 0.439 |
| 080 | **0.998** | 0.959 | 0.403 |
| 081 | 0.990 | 0.928 | **1.000** |
| 082 | **0.957** | 0.854 | 0.858 |
| 083 | **0.758** | 0.593 | 0.547 |
| 084 | **0.825** | 0.732 | 0.557 |
| 085 | **0.793** | 0.378 | 0.563 |
| 086 | **0.766** | 0.737 | 0.605 |
| 087 | **0.795** | 0.734 | 0.307 |
| 088 | **0.929** | 0.689 | 0.808 |
| 089 | **0.727** | 0.473 | 0.717 |
| 090 | **0.991** | 0.948 | 0.985 |
| 091 | 0.705 | 0.553 | **0.761** |
| 092 | **0.725** | 0.619 | 0.589 |
| 093 | 0.910 | **0.939** | 0.886 |
| 094 | **0.963** | 0.484 | 0.505 |
| 095 | **0.962** | 0.455 | 0.456 |
| 096 | **0.990** | 0.303 | 0.345 |
| 097 | **0.858** | 0.240 | 0.352 |
| 098 | **0.939** | 0.636 | 0.611 |
| 099 | **0.976** | 0.742 | 0.940 |
| 100 | **0.833** | 0.758 | 0.763 |
| 101 | **0.990** | 0.252 | 0.177 |
| 102 | **0.893** | 0.543 | 0.630 |
| 103 | **0.918** | 0.867 | 0.832 |
| 104 | **0.797** | 0.517 | 0.516 |
| 105 | **0.569** | 0.544 | 0.493 |
| 106 | 0.848 | 0.466 | **0.858** |
| 107 | **0.693** | 0.368 | 0.230 |
| 108 | **0.650** | 0.486 | 0.593 |
| 109 | **0.664** | 0.465 | 0.638 |
| 110 | **0.943** | 0.818 | 0.896 |
| 111 | **0.570** | 0.134 | 0.150 |
| 112 | **0.914** | 0.568 | 0.476 |
| 113 | **0.659** | 0.453 | 0.432 |
| 114 | **0.580** | 0.231 | 0.506 |
| 115 | **0.800** | 0.715 | 0.448 |
| 116 | **0.931** | 0.645 | 0.084 |
| 117 | **0.581** | 0.263 | 0.296 |
| 118 | **0.785** | 0.721 | 0.520 |
| 119 | 0.510 | 0.445 | **0.538** |
| 120 | **0.983** | 0.578 | 0.582 |
| 121 | 0.768 | **0.800** | 0.456 |
| 122 | 0.611 | **0.731** | 0.698 |
| 123 | **0.558** | 0.309 | 0.354 |
| 124 | **0.851** | 0.572 | 0.801 |
| 125 | **0.764** | 0.708 | 0.469 |
| 126 | **0.922** | 0.679 | 0.768 |
| 127 | **0.792** | 0.525 | 0.626 |
| 128 | 0.662 | **0.684** | 0.592 |
| 129 | 0.582 | 0.588 | **0.645** |
| 130 | **0.947** | 0.816 | 0.897 |
| 131 | 0.505 | **0.585** | 0.550 |
| 132 | **0.795** | 0.383 | 0.328 |
| 133 | **0.688** | 0.458 | 0.552 |
| 134 | **0.877** | 0.425 | 0.365 |
| 135 | **0.917** | 0.669 | 0.735 |
| 136 | **0.763** | 0.606 | 0.756 |
| 137 | **0.969** | 0.810 | 0.103 |
| 138 | **0.830** | 0.726 | 0.768 |
| 139 | **0.954** | 0.168 | 0.916 |
| 140 | **0.639** | 0.494 | 0.490 |
| 141 | **0.714** | 0.476 | 0.451 |
| 142 | **0.945** | 0.391 | 0.362 |
| 143 | **0.996** | 0.720 | 0.681 |
| 144 | 0.994 | 0.977 | **0.995** |
| 145 | **0.895** | 0.856 | 0.878 |
| 146 | **0.798** | 0.728 | 0.738 |
| 147 | **0.647** | 0.531 | 0.327 |
| 148 | **0.649** | 0.413 | 0.371 |
| 149 | **0.481** | 0.353 | 0.356 |
| 150 | **0.822** | 0.705 | 0.685 |
| 151 | **0.853** | 0.183 | 0.800 |

watershed transform. We also intend to investigate FLIM in the context of different applications for CNNs.

## ACKNOWLEDGMENTS

This research was funded by São Paulo Research Foundation (FAPESP) (2014/12236-1 and 2019/10705-8), Nacional Council for Scientific and Technological Development (CNPq) (303808/2018-7), Petróleo Brasileiro S.A. (PETROBRAS) and Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP), Grant Numbers 4600556376 and 4600583791.

## REFERENCES

[1] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1 – 15, 2018.

[2] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[3] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[4] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[5] R. Lotufo and A. Falcao, *The Ordered Queue and the Optimality of the Watershed Approaches*. Boston, MA: Springer US, 2000, pp. 341–350.

[6] I. E. de Souza and A. X. Falcão, "Learning cnn filters from user-drawn image markers for coconut-tree image classification," *IEEE Geoscience and Remote Sensing Letters*, to appear.

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

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks

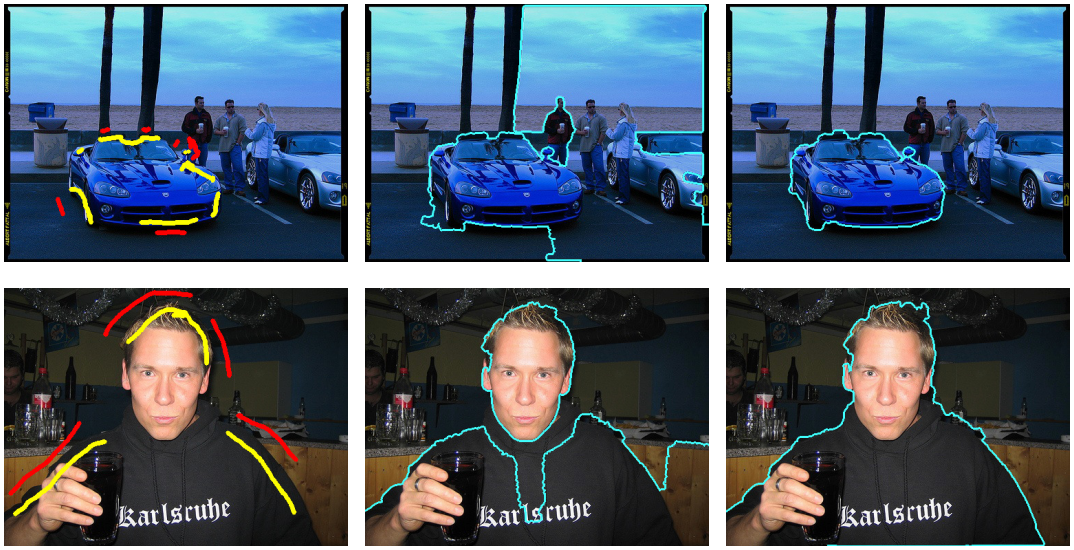



Fig. 4. Marker selection suitable for FLIM-based filter estimation (left), watershed delineation using the Lab-color-based gradient (center), and using the FLIM-based features (right). Both models have a single layer with 152 and 97 filters of 5×5 pixels for the top and bottom images, respectively.

for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[10] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[11] Y. Xian, S. Choudhury, Y. He, B. Schiele, and Z. Akata, "Semantic projection network for zero-and few-label semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8256–8265.

[12] S. Mahadevan, P. Voigtlaender, and B. Leibe, "Iteratively trained interactive segmentation," *arXiv preprint arXiv:1805.04398*, 2018.

[13] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang, "Deep interactive object selection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[14] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017.

[15] W.-D. Jang and C.-S. Kim, "Interactive image segmentation via backpropagating refinement scheme," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5297–5306.

[16] K. Sofiiuk, I. Petrov, O. Barinova, and A. Konushin, "F-brs: Rethinking backpropagating refinement for interactive segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[17] T. Kontogianni, M. Gygli, J. Uijlings, and V. Ferrari, "Continuous adaptation for interactive object segmentation by learning from corrections," *arXiv preprint arXiv:1911.12709*, 2019.

[18] T. Spina, P. A. V. de Miranda, and A. X. Falcão, "Intelligent understanding of user interaction in image segmentation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 02, p. 1265001, 2012.

[19] 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*, 2015, pp. 3431–3440.

[20] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive annotation: A deep active learning framework for biomedical image segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 399–407.

[21] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal, "Reinforced active learning for image segmentation," *arXiv preprint arXiv:2002.06583*, vol. arXiv preprint arXiv:2008.03549, 2020.

[22] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[24] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *arXiv preprint arXiv:2001.05566*, 2020.

[25] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman, "Geodesic star convexity for interactive image segmentation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3129–3136.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and T. scikit-image contributors, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.