Image Analysis Project

Prof. Alexandre Xavier Falcão

September 19, 2023

1 Overview

This project consists of partially implementing and evaluating the following sequence of operations: (1) median filtering to reduce image noise; (2) a convolutional neural network (CNN) that estimates an object saliency map; and (3) object detection/delineation using the properties of the filtered image and estimated saliency map. The CNN's encoder follows a given architecture provided in a JSON file. Its weights are estimated from markers that the user draws on discriminative regions of very few training images (e.g., 5) – i.e., the user defines object and background markers using the FLIMBuilder tool provided in libmo445. Each marker generates a dataset of image patches centered at marker pixels. By clustering this dataset, the cluster centers define the kernels, and each kernel receives the label of its corresponding marker. We will adopt a variant of the method Feature Learning from Image Markers (FLIM) proposed in [1]. The decoder has a single layer with a point-wise convolution followed by an activation function. The decoder's weights may be fixed, based on the kernels' labels, or adaptive, similar to the approach presented in [2]. This decoder can estimate a saliency map at the output of any encoder's layer. We will evaluate the results of object detection and delineation from those outputs. Object detection will follow the approach in [2] – it binarizes the saliency map and post-processes the mask. Object delineation will use the saliency map to estimate markers for the Image Foresting Transform (IFT) [3]. We will compare two IFT-based object delineation methods: watershed transform [4] and dynamic trees [5, 6]. Object delineation using watershed transform and CNN features extracted from image markers was presented in [7]. In our case, we will explore the saliency map rather than activations from the FLIM-based CNN.

Your first task is then to read references [1, 2, 3, 5, 7] in this order. You will find the pdf files of [3, 5] in the folder "articles". The links to the pdf files of the remaining references are provided in Section References.

2 Definition of the problem

The project consists of building sequence (1)-(3) of operations for <mark>detecting</mark> and <mark>delineating parasite eggs in optical microscopy images</mark>, given the constraint of using a minimum of training images. There are two folders, "images" and "truelabels" – the former contains the original

images with/without parasite eggs, and the latter contains their manually delineated masks. The project involves drawing bounding boxes around the detected objects and delineating their segmentation masks. Modify the codes to report effectiveness measures, such as f-score for delineation and intersection over union for detection.

3   Background on FLIM-based CNNs

This section provides definitions and insights to better understand FLIM-based CNNs.

3.1   Multi-channel and multi-dimensional images.

Let $\hat{I} = (D, \mathbf{I})$ be a multi-channel image such that $\mathbf{I}$ assigns an **element feature vector** $\mathbf{I}(p) = (I_1(p), I_2(p), \ldots, I_m(p)) \in \Re^m$ with $m \geq 1$ scalar values (channels) to every *spel* p (space element) of a multi-dimensional image domain $D \subset Z^n$, $n \geq 1$ [1]. This definition is general enough for most types of images, such as remote sensing ($n = 2$, $m \geq 1$), digital photography ($n = 2$, $m = 3$), and volumetric medical images ($n = 3$, $m = 1$). Images may also be acquired over time, forming a sequence of natural (e.g., digital video) or volumetric medical images. In such cases, for simplicity, time is the n-th dimension, being a spel a space-time element with $n - 1$ spatial dimensions and 1 temporal dimension.

3.2   Adjacency relations and image patches.

An adjacency relation $\mathcal{A}$ is a binary relation between spels $(p, q) \in D \times D$. One can define $\mathcal{A}$ in many different ways, but we focus on reflexive and translation-invariant adjacency relations. For instance,

$$\mathcal{A} : \{(p, q) \in D \times D \mid \|q - p\| \leq \rho\} \tag{1}$$

defines a hyper-sphere centered at p with radius $\rho \geq 0$. Another example in the case of 2D images, the most commonly used for deep learning, is

$$\mathcal{A} : \{(p, q) \in D \times D \mid |x_q - x_p| \leq d_x, |y_q - y_p| \leq d_y\} \tag{2}$$

for $p = (x_p, y_p) \in D \subset Z^2$ and displacements $d_x \geq 0$, $d_y \geq 0$. In any case, $\mathcal{A}(p)$ is a set of spels $q \in D$ that are adjacent to p and $|\mathcal{A}(p)| = k_\mathcal{A} \geq 1$ is fixed for any $p \in D$ [2].

For any given spel $p \in D$, a **patch** is the subimage $\hat{P}(p) = (\mathcal{A}(p), \mathbf{I})$ centered at p in which every adjacent spel $q \in \mathcal{A}(p)$ is represented by its element feature vector $\mathbf{I}(q)$.

---

[1] An image of dimension $D \subset Z^1$ is degenerated into a signal.

[2] One can think of D as an extended image domain with zeros outside it, such that all adjacent spels of $p \in D$ will be inside it – the so called zero-padding operation.

3.3 Linear filters, convolution, and insightful interpretations

A filter (kernel) $\hat{F} = (\mathcal{A}, \mathbf{F})$ is a "moving sub-image" with the same shape of a patch, but fixed values $\mathbf{F}(q - p) \in \Re^m$ for each $q \in \mathcal{A}(p)$ and any filter position p. The convolution [3] between $\hat{I}$ and $\hat{F}$ produces a filtered image $\hat{J} = (D, J)$, where

$$
J(p) = \sum_{k=1}^{k_{\mathcal{A}}} \langle \mathbf{I}(q_k), \mathbf{F}(q_k - p) \rangle, \tag{3}
$$

$$
\langle \mathbf{I}(q_k), \mathbf{F}(q_k - p) \rangle = \sum_{j=1}^{m} I_j(q_k) F_j(q_k - p),
$$

for $p \in D$ and $\mathcal{A}(p) = \{q_1, q_2, \ldots, q_{k_{\mathcal{A}}}\}$. When image $\hat{I}$ is filtered by a set (bank) with m' filters, each filtered image represents a new channel, and image $\hat{J}$ is a multi-channel image $(D, \mathbf{J})$ with $\mathbf{J}(p) = (J_1(p), J_2(p), \ldots, J_{m'}(p)) \in \Re^{m'}$.

A desired simplification for Equation 3 requires interpreting patches and filters as vectors in $\Re^{m \times k_{\mathcal{A}}}$. Let a patch $\hat{P}(p)$ be interpreted as a **local feature vector** $\widetilde{\mathbf{P}}(p)$ resulting from the concatenation $\mathbf{I}(q_1) \oplus \mathbf{I}(q_2) \oplus \ldots \oplus \mathbf{I}(q_{k_{\mathcal{A}}})$ for $\mathcal{A}(p) = \{q_1, q_2, \ldots, q_{k_{\mathcal{A}}}\}$. Similarly, a filter $\hat{F} = (\mathcal{A}(p), \mathbf{F})$ can be interpreted by its weight vector $\widetilde{\mathbf{F}}$ resulting from the concatenation $\mathbf{F}(q_1 - p) \oplus \mathbf{F}(q_2 - p) \oplus \ldots \oplus \mathbf{F}(q_{k_{\mathcal{A}}} - p)$. By that, convolution at every spel p becomes the dot product between a patch extracted at p and the filter.

$$
J(p) = \langle \widetilde{\mathbf{P}}(p), \widetilde{\mathbf{F}} \rangle. \tag{4}
$$

Equation 4 has a meaningful geometric interpretation when filtering images for feature extraction. Filter $\widetilde{\mathbf{F}}$ can be interpreted as a vector orthogonal to a hyperplane passing through the origin of $\Re^{m \times k_{\mathcal{A}}}$ and $\widetilde{\mathbf{P}}(p)$ is a point in $\Re^{m \times k_{\mathcal{A}}}$ (Figure 1). When the norm $|\widetilde{\mathbf{F}}| = 1$, $J(p)$ is the distance between $\widetilde{\mathbf{P}}(p)$ and the hyperplane of $\widetilde{\mathbf{F}}$ – it may be positive or negative depending on which side of the hyperplane the point $\widetilde{\mathbf{P}}(p)$ is. When $\widetilde{\mathbf{F}}$ does not have a unit norm, $J(p)$ is amplified by its magnitude. The convolution between an image and a filter slides the filter over the image and computes the signed distance $J(p)$ for every spel $p \in D$.

Another insightful interpretation appears when the filtered image $\hat{J} = (D, J)$ is further transformed by an **activation function** ϕ defining image $\hat{K} = (D, K)$ where $K(p) = \phi(J(p))$. A commonly used example is the Rectified Linear Unit (ReLU) operation

$$
K(p) = \max\{0, J(p)\}. \tag{5}
$$

In this case, convolution followed by activation is equivalent to interpreting $\widetilde{\mathbf{F}}$ as the weight vector of an **artificial neuron** with activation function ϕ placed at every spel p and **receptive field** $\mathcal{A}(p)$ with weights $\mathbf{F}(q - p)$ for $q \in \mathcal{A}(p)$.

We may interpret the activation $K(p)$ as a similarity between the patch $\widetilde{\mathbf{P}}(p)$ and the local visual pattern represented by $\widetilde{\mathbf{F}}$. However, without normalization, farther $\widetilde{\mathbf{P}}(p)$ is to the

---

[3]The original concept is simplified here to preserve the image domain and assume the adjacency relation is already reflected in each axis.
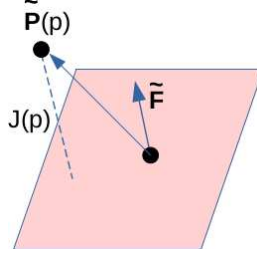
Figure 1: Geometric interpretation of the convolution operation at a spel p.

positive side of the hyperplane more similar it is to the pattern $\widetilde{\mathbf{F}}$, which does not make sense when measuring similarity by a dot product. Such normalization can be obtained from the image markers, as explained next.

3.4 Marker-based image normalization and bias.

One can use marker spel values for image normalization before convolution and activation by drawing strokes on discriminative regions of a few representative images.

Let $\mathcal{M}(\hat{I})$ be the set of marker spels drawn on image $\hat{I}$, $\mathcal{I}$ be a small set of images annotated by markers, and $\mathcal{M}$ be the union $\cup_{\hat{I} \in \mathcal{I}} \mathcal{M}(\hat{I})$ of all marker sets. We can transform any image $\hat{I}$ into a normalized image $\hat{N} = (D, \mathbf{N})$, where $\mathbf{N}(p) = (N_1(p), N_2(p), \ldots, N_m(p)) \in \Re^m$ with

$$N_j(p) \;\;=\;\; \frac{I_j(p) - \mu_j}{\sigma_j + \epsilon}, \tag{6}$$

$$\mu_j \;\;=\;\; \frac{1}{|\mathcal{M}|} \sum_{\hat{I} \in \mathcal{I}, p \in \mathcal{M}(\hat{I})} I_j(p), \tag{7}$$

$$\sigma_j^2 \;\;=\;\; \frac{1}{|\mathcal{M}|} \sum_{\hat{I} \in \mathcal{I}, p \in \mathcal{M}(\hat{I})} \left(I_j(p) - \mu_j \right)^2, \tag{8}$$

for a small $\epsilon > 0$.

One may see the convolution between the normalized image $\hat{N}$ and a filter $\hat{F}$ as equivalent to

- convolving $\hat{I}$ and a filter $\hat{F}' = (\mathcal{A}, \mathbf{F}')$, with $\mathbf{F}'(q - p) \in \Re^m$,

$$F_j'(q - p) \;\;=\;\; \frac{F_j(q - p)}{\sigma_j + \epsilon}, \text{ and} \tag{9}$$

- adding a bias $b \in \Re$ with

$$b \;\;=\;\; -\sum_{q \in \mathcal{A}(p)} \sum_{j=1}^{m} \frac{\mu_j F_j(q - p)}{\sigma_j + \epsilon}. \tag{10}$$

4

Another alternative is to apply Z-score normalization to the patch vectors $\widetilde{\mathbf{P}}(p) \in \Re^{m \times k_{\mathcal{A}}}$ using their mean $\widetilde{\mu} \in \Re^{m \times k_{\mathcal{A}}}$ and standard deviation $\widetilde{\sigma} \in \Re^{m \times k_{\mathcal{A}}}$ vectors. By forcing $|\widetilde{\mathbf{F}}| = 1$, the new filter coeficients and bias can be computed as follows.

$$\widetilde{F}'_i \;=\; \frac{\widetilde{F}_i}{\sigma_i + \epsilon} \tag{11}$$

$$b \;=\; -\sum_{i=1}^{m \times k_{\mathcal{A}}} \mu_i \widetilde{F}'_i \tag{12}$$

where $\widetilde{\mathbf{F}} = (\widetilde{F}_1, \widetilde{F}_2, \ldots, \widetilde{F}_{m \times k_{\mathcal{A}}})$, $\widetilde{\mathbf{F}}' = (\widetilde{F}'_1, \widetilde{F}'_2, \ldots, \widetilde{F}'_{m \times k_{\mathcal{A}}})$, $\widetilde{\mu} = (\mu_1, \mu_2, \ldots, \mu_{m \times k_{\mathcal{A}}})$, $\widetilde{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_{m \times k_{\mathcal{A}}})$.

By that, marker-based normalization aims to avoid bias computation after convolution. On the other hand, if the user draws additional markers, Equations 9 and 10 allow to eliminate marker-based normalization, merge filters from multiple marker sets, and then use their individual biases after convolution.

3.5 Pooling.

After marker-based normalization, convolution, and activation, pooling can aggregate nearby activation values of patterns related to the same object/category. For simplicity, we focus on max-pooling. Assuming the convolution uses a filter bank with m' filters, after activation, image $\hat{K} = (D, \mathbf{K})$ has m' channels (i.e., $\mathbf{K}(p) \in \Re^{m'}$). Max-pooling defines a new image $\hat{P} = (D, \mathbf{P})$, where $\mathbf{P}(p) = (P_1(p), P_2(p), \ldots, P_{m'}(p)) \in \Re^{m'}$ with

$$P_j(p) \;=\; \max_{q \in \mathcal{B}(p)} \{K_j(q)\}, \tag{13}$$

for $j = 1, 2, \ldots, m'$ and an adjacency relation $\mathcal{B}$.

It is usually desired to reduce the domain D in each block, which can be done in pooling by a *stride* factor per spatial direction (distance between spels in each direction).

3.6 Convolutional blocks and feature extraction.

A *convolutional block* is a set of image operations such as marker-based normalization, convolution, activation, and pooling. Each image operation is referred to as a layer, and a block may define a sequence of layers or combine the outputs of multiple layers by adding their corresponding spel values or concatenating their channels. A block can also contain multiple layers of the same type, reducing or increasing the number of channels.

A *convolutional feature extractor* (encoder) is a sequence of such blocks, transforming multi-channel images into other images with multiple channels. However, such transformations have a purpose – they aim to separate activations from different classes/objects into distinct (spel, channel) coordinates. For image classification, **flattening** transforms an output image $\hat{O}$ into a **global feature vector** $\mathbf{x}(\hat{O}) = (x_1, x_2, \ldots, x_n) \in \Re^n$, such that x_i represents the activation

of a spel in a channel of $\hat{O}$. Let $\lambda(\hat{O}) \in \{1, 2, \ldots, c\}$ be the true-label of image $\hat{O}$, a *predictor* should assign $L(\hat{O}) = \lambda(\hat{O})$ for correctness. If $\mathbf{x}(\hat{O})$ for every $\hat{O}$ with true-label $\lambda(\hat{O})$ falls in a subspace $\Re^k$, $k < n$, different from the subspace of samples from other labels, then a single hyperplane per label with orthogonal vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)$, such that $u_i = 1$ for each axis in $\Re^k$ and $u_i = 0$ otherwise, would suffice to predict $L(\hat{O}) = \lambda(\hat{O})$.

3.7   Convolutional Neural Networks

A CNN for image classification is a sequence: convolutional feature extraction, flattening, and prediction. The predictor is usually a multi-layer perceptron, but it could be a support vector machine. For object (instance or semantic) segmentation, each spel p belongs to an object among c objects in $\{1, 2, \ldots, c\}$. Segmentation consists of two coupled operations – object detection and object delineation – with the outputs of the first convolutional blocks more suitable for delineation and the outputs of the last convolutional blocks more suitable for detection. Therefore, a *decoder* uses image operations, such as upsampling, concatenation, transpose convolution, and activation, to combine those outputs into a multi-channel image with the same size of the input for a last decoding block with one *point-wise convolution* per object.

A point-wise convolution is essentially a weighted average of the image channels with positive weights indicating that the channel is relevant for the object and negative weights indicating the opposite. This last block creates one *salience map per object* – a map whose activations should be higher to the object's spels than to the background. A *softmax operation* then compares the salience values of each spel $p \in D$ in all maps and assigns the label $L(p) \in \{1, 2, \ldots, c\}$ of the object with the highest salience value to solve segmentation. Therefore, it is expected that the encoder, together with the previous decoder blocks, create a multi-channel image in which distinct objects present higher activations in different channels.

3.8   Feature Learning from Image Markers (FLIM)

The main steps in a FLIM encoder are illustrated in Figure 2. Initially, a few representative images per class/object are selected (Figure 2a) for marker drawing. The user then draws scribbles on discriminative regions (Figures 2b and 3a) – i.e., parts that distinguish classes/objects. The interior and exterior of the objects are usually used for segmentation [7] while texture boundaries are recommended for classification. However, this step is still subjective and under investigation.

Filter estimation (Figure 2c) for a given convolutional block (Figure 2d) relies on a dataset consisting of patches with the exact shape of the filters centered at marker spels (Figure 3a) and extracted from the input images of the block – i.e., original or activation images as shown in (c)-(d) of Figure 2. In FLIM-based methods, the patches are extracted from attention regions as indicated by the expert, and the cluster centers derive the convolutional filters (Figure 3b). The methods apply marker-based normalization and patch clustering. The cluster centers are forced to the unit norm and then used as filters in the convolutional block [1]. A higher number of clusters than desired is usually obtained by fixing the number of clusters per marker

6

to guarantee groups from all attention regions. Principal Component Analysis then reduces the number of candidates.

The convolution with cluster centers activates regions on the positive side of the filter's hyperplane, eliminating the regions on its negative side (Figure 3b). Each filter is expected to separate discriminative class/object parts in different image channels (Figure 3c). For a given encoder architecture, markers can be projected at the input of each block for filter learning with no user interference [1], as illustrated in (c)-(d) of Figure 2. One can also inspect the block's activations as a simple filter evaluation procedure (Figure 2e).

In this project, however, we will adopt a given number of kernels per marker since each feature point will correspond to one kernel. We will also avoid clustering along the layers. For kernel evaluation, you may normalize in [0,255] the activation channels for visualization and/or execute the decoder to analyze the resulting object saliency map of any given layer.
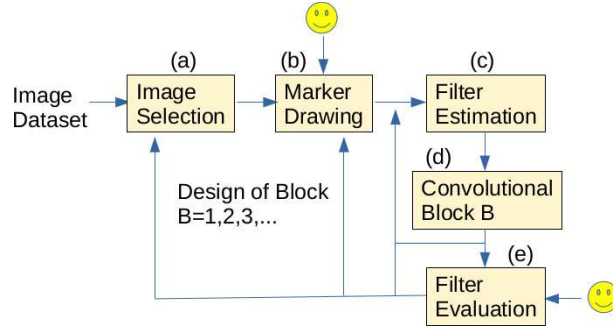
Figure 2: The main steps when designing a FLIM encoder.

4   Detection and delineation of parasite eggs

Figure 4 shows a sequence of operations for detecting and delineating parasite eggs. The median filter (code preproc.c) receives the adjacency radius parameter. Evaluate the impact of this operation for distinct parameter values. Select a few training images (e.g., 5) and draw markers (disks or strokes) inside and outside the objects. Including examples with distinct egg appearances is desirable to represent the problem better. Draw object and background markers with distinct labels in FLIMBuilder. The markers should represent discriminative object regions and background regions. The code bag_of_feature_points.c estimates feature points from the drawn markers. Evaluate it for one and a few points per marker. After drawing markers in all training images and creating a bag of feature points, use create_layer_models.c to create one model per image. At this point, it is possible to keep the pipeline's loop, using encode_layer.c, with one model per image to verify the results for the training images only. Another option is to merge all models of the current layer into a single consolidated model with merge_layer_models.c. After merging the models, use encode_merged_layer.c to proceed with the pipeline's loop. At any iteration of the pipeline's loop, it is possible to use decode_layer.c
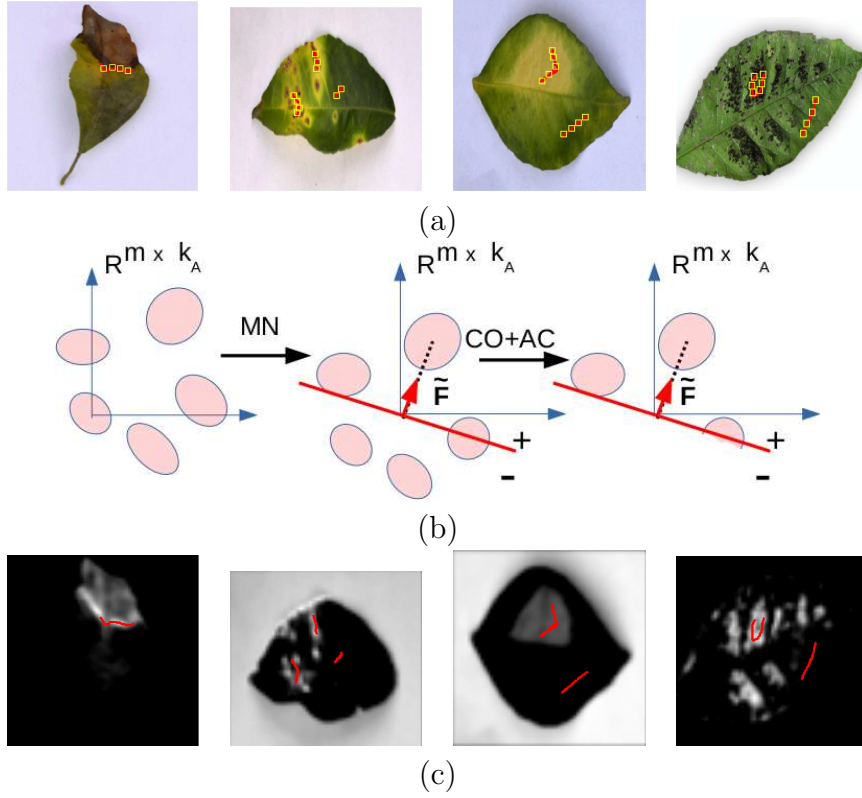
(a)

(b)

(c)

Figure 3: (a) Scribbles (red) are drawn on discriminative parts of four classes of leaf diseases, and patch feature vectors $\widetilde{\mathbf{P}}(p) \in \Re^{m \times k_{\mathcal{A}}}$ (yellow) are extracted from their spels p. (b) Marker-based normalization (MN) centralizes the patch distribution in $\Re^{m \times k_{\mathcal{A}}}$ with feature normalization. Cluster centers derive the filters $\widetilde{\mathbf{F}} \in \Re^{m \times k_{\mathcal{A}}}$ (red arrow). Convolution (CO) with $\widetilde{\mathbf{F}}$ followed by ReLU activation (AC) enhances regions on the positive side of the filter's hyperplane (red line). (c) Examples of activation images obtained from the drawn markers.

to create a saliency map, detection.c to transform saliency maps into detected regions, and delineation.c to delineate the boundary of the detected objects. In decode_layer.c, evaluate different activation functions and adaptive decoders. In detection.c, incorporate the true-label image and estimate intersection-over-union (IoU). In delineation.c, incorporate the true-label image and estimate the f-score.

Some of the above codes will contain empty functions to be completed before testing the pipeline. Note that when using the consolidated models of each layer, the pipeline will also execute on the test images. Evaluate different architecture parameters for convolution with/without dilation, patch sizes, pooling types (no_pool, max_pool, or avg_pool), and strides.
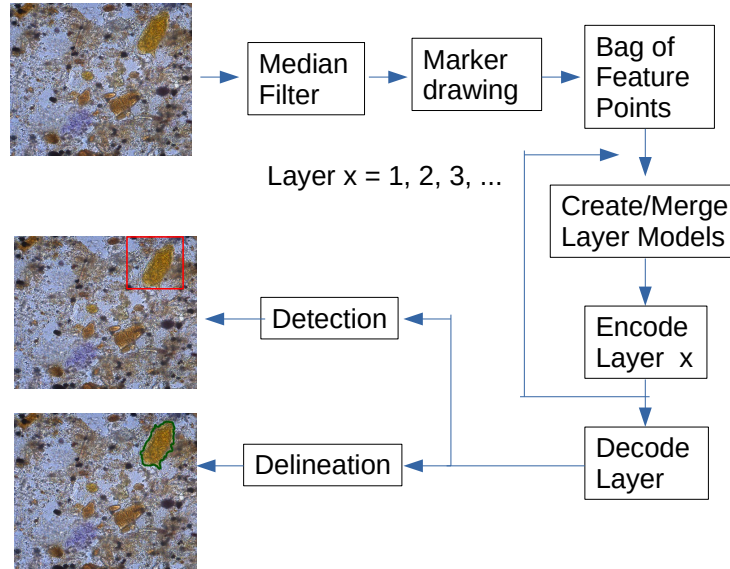
Figure 4: Sequence of operations for parasite egg detection and delineation

5   Questions to be answered

(2,0) How does the number of training images and feature points per marker affect the results for a given architecture?

(1,0) What is the impact of the median filter in the process, especially in object delineation?

(1,0) For a given architecture, analyze the output activations and saliency maps. What is it observed? How deep can the encoder be? Should it have different depths for detection and delineation, respectively?

(2,0) What is the best architecture among the evaluated ones for the problem?

(1,5) How do the decoder options compare?

(0,5) What is the best activation function to create the saliency maps: ReLU, Sigmoid, or another type of activation function?

(2,0) The delineation code requires inner and outer seed estimation followed by a watershed transform or a dynamic-tree algorithm. Adjust the parameters for seed estimation and compare both solutions. Which one is the best for the problem?

6 Evaluation criteria

Evaluation will consider the quality of the report in answering the above questions, which totalize a score in [0,10]. The students will then be graded as follows.

- grade A for average score in [8.5,10];

- grade B for average score in [7.0,8.5);

- grade C for average score in [5.0,7.0);

- grade D for average score in [0.0,5.0);

The reports should have a maximum of 20 pages with letter-size 11pt, including figures, tables, graphics, and references. It should present the following organization.

- Cover page: provide the name of the discipline, name of the students, academic identification numbers (RA), and delivery date, followed by a summary of the main results.

- Subsequent pages: present the literature that has been studied to implement the project, its difficulties, given solutions, implemented algorithms, qualitative and quantitative experimental results that answer the above questions, and discussion.

- Final page: present a conclusion about what has been accomplished with the task and provide suggestions to improve it.

References

[1] I. 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, doi: 10.1109/LGRS.2020.3020098, 2020, https://arxiv.org/pdf/2008.03549.pdf.

[2] Leonardo de Melo Joao and Azael de Melo e Sousa and Bianca Martins dos Santos and Silvio Jamil Ferzoli Guimaraes and Jancarlo Ferreira Gomes and Ewa Kijak and Alexandre Xavier Falcao. A Flyweight CNN with Adaptive Decoder for Schistosoma mansoni Egg Detection, arXiv 2306.14840, 2023, https://doi.org/10.48550/arXiv.2306.14840.

[3] A.X. Falcão, J. Stolfi and R.A. Lotufo. The Image Foresting Transform: Theory, Algorithms, and Applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, ISSN 0162-8828, vol. 26, No. 1, pp. 19–29, doi: 10.1109/TPAMI.2004.1261076, January, 2004.

[4] R.A. Lotufo and A.X. Falcão. The Ordered Queue and the Optimality of the Watershed Approaches. *5th Intl. Symp. on Mathematical Morphology*, Mathematical Morphology and Its Applications to Image and Signal Processing (Computational Imaging, vol. 18, 456p, ISBN 0-7923-7862-8), John Goutsias (Editor), Luc Vincent (Editor), Dan S. Bloomberg (Editor), Kluwer Academic, (Palo Alto, CA, pp. 341-350, 2000.

[5] A.X. Falcão and J. Bragantini. The Role of Optimum Connectivity in Image Segmentation: Can the Algorithm Learn Object Information During the Process? Proc. of 21st Discrete Geometry for Computer Imagery (DGCI). Couprie M., Cousty J., Kenmochi Y., Mustafa N. (eds), doi 10.1007/978-3-030-14085-4_15, LNCS 11414, pp. 180-194, March 2019.

[6] J. Bragantini, S.B. Martins, C. Castelo-Fernández, and A.X. Falcão. Graph-based Image Segmentation using Dynamic Trees. *23rd Iberoamerican Congress on Pattern Recognition*, CIARP 2018: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, doi 10.1007/978-3-030-13469-3_55, LNCS 11401, Madrid, Spain, pp. 470–478, 2019.

[7] I. de Souza, B.C. Benato, and A.X. Falcão. Feature Learning from Image Markers for Object Delineation. *33rd SIBGRAPI: Conference on Graphics, Patterns and Images*, doi: 10.1109/SIBGRAPI51738.2020.00024, Porto de Galinhas, PE, pp. 116-123, 2020, http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2020/09.30.14.16/doc/76.pdf?metadatarepository=sid.inpe.br/sibgrapi/2020/09.30.14.16.57&mirror=sid.inpe.br/banon/2001/03.30.15.38.24.