Kato *et al. BMC Bioinformatics* (2015) 16:316
DOI 10.1186/s12859-015-0739-1

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Segmental HOG: new descriptor for glomerulus detection in kidney microscopy image

Tsuyoshi Kato[1,2*], Raissa Relator[1], Hayliang Ngouv[1], Yoshihiro Hirohashi[1], Osamu Takaki[3], Tetsuhiro Kakimoto[4] and Kinya Okada[4]

Abstract

Background: The detection of the glomeruli is a key step in the histopathological evaluation of microscopic images of the kidneys. However, the task of automatic detection of the glomeruli poses challenges owing to the differences in their sizes and shapes in renal sections as well as the extensive variations in their intensities due to heterogeneity in immunohistochemistry staining.
Although the rectangular histogram of oriented gradients (Rectangular HOG) is a widely recognized powerful descriptor for general object detection, it shows many false positives owing to the aforementioned difficulties in the context of glomeruli detection.

Results: A new descriptor referred to as Segmental HOG was developed to perform a comprehensive detection of hundreds of glomeruli in images of whole kidney sections. The new descriptor possesses flexible blocks that can be adaptively fitted to input images in order to acquire robustness for the detection of the glomeruli. Moreover, the novel segmentation technique employed herewith generates high-quality segmentation outputs, and the algorithm is assured to converge to an optimal solution. Consequently, experiments using real-world image data revealed that Segmental HOG achieved significant improvements in detection performance compared to Rectangular HOG.

Conclusion: The proposed descriptor for glomeruli detection presents promising results, and it is expected to be useful in pathological evaluation.

Keywords: Microscopy image analysis, Glomerulus detection, Computer vision, Support vector machine, Dynamic programming, Glomerular injury marker, Desmin immunostaining

Background

The glomeruli in the kidneys act as a filtration barrier that retains higher molecular weight proteins in blood circulation. In various renal diseases, the glomerular filtration barrier can be damaged, resulting in protein leakage into urine, known as proteinuria. Therefore, the pathological changes in renal glomeruli of animal disease models can provide important information in screening compounds that target such diseases.

Our goal was to perform high-throughput detection of the glomeruli in highly enlarged microscopy images of animal disease models, whose sizes run up to the order of 10^8 pixels. Although there are existing studies about the automatic analysis of the glomeruli in microscopic images of the kidneys [1, 2], the target images in these studies were obtained from human biopsy samples with relatively small sizes; therefore, they are not suitable for our purpose.

Compared to general object detection tasks, there are two particular obstacles in the case of glomeruli detection. The first obstacle arises from the non-rigid sizes and

*Correspondence: katotsu@cs.gunma-u.ac.jp
[1]Faculty of Science and Engineering, Gunma University, Kiryu-shi, Gunma 376–8515, Japan
[2]Center for Informational Biology, Ochanomizu University, Bunkyo-ku, Tokyo 112-8610, Japan
Full list of author information is available at the end of the article

shapes of the targets in the images. Indeed, the glomeruli have a fixed size *in vivo*, although they swell to some extent in unfavorable situations such as hypertension [3] and diabetes [4]. In addition, the sizes of the glomeruli in a whole-kidney-section image could vary depending on where the cross-section was taken. The shape of the glomeruli is mostly spherical, making the boundaries circular. To obtain the boundaries, one might try to fit an ellipse to each glomerulus. However, this approach yields large estimation errors because each glomerulus is deformed to some extent.

The second difficulty arising in the glomerulus detection task is the high variation in staining intensity. On histological evaluation, immunohistochemistry is usually used to demonstrate the distribution and location of proteins in tissue sections. In our target images, sections were immunostained for desmin, a known glomerular injury marker. Therefore, some glomeruli are stained and some are not. As many glomeruli are partly stained, resulting in heterogeneously stained glomeruli, detection is more complicated. Furthermore, the stained tissues in the kidneys are not only from the glomeruli but also from other tissues such as the blood vessels.

To check the existence of a glomerulus at each location in a whole-kidney-section image, the sliding window technique is employed [5–8]. Using this procedure, a frame goes over the input image to check for the target object at every possible location; then, a descriptor of the sub-image is extracted.

Rectangular HOG (R-HOG) [9], a widely used and recognized efficient descriptor for object detection in the field of computer vision, is a potentially suitable candidate descriptor for glomeruli. It has the capacity to capture information about the magnitude of the gradients in the image. Therefore, this descriptor is robust to the change in intensities caused by the heterogeneity of the stained levels. Glomeruli are known to be composed of tightly packed cells, resulting in high gradients on images. Thus, a natural approach would be to use the magnitude of these gradients as features of the glomeruli. However, although we have previously attempted to directly exploit this attribute, we found the detection performance to be poor, resulting in many false positives and low recall. In addition to the magnitude of the gradients, their directions are also important to distinguish the glomeruli from the other tissues. Using R-HOG descriptors obtained from both the magnitude and the direction of the gradients, glomeruli detection performance results in recall values high enough to be useful for pathological evaluation. However, it appeared that R-HOG still has a considerable amount of false positives [10–12].

The high number of false positives in previous studies [10–12] can be ascribed to the condition that the standard HOG such as the R-HOG has a rigid block division. Owing to this rigidity, there are instances when a block is inside the glomerular area in a case and outside in another. Thus, extracted features from each block contain large differences, and robustness for the deformations of glomeruli is lost. Although there are several other known local descriptors such as scale-invariant feature transform (SIFT) image features [13], Haar-like features [8], and local binary patterns (LBP) [14], these do not possess a solution to be robust for deformed glomeruli for similar reasons.

In this study, we introduce flexible block division to the HOG descriptor to improve the detection performance and to reduce the number of false positives. A new feature, which we refer to as the Segmental HOG (S-HOG) descriptor, has been proposed for glomerulus detection. The block division of S-HOG is based on the estimated boundary of the glomerulus that is obtained via a segmentation algorithm, which has also been developed in this study. This renders the division of blocks to be more adaptable than the rigid block division of R-HOG, and allows feature vectors to clearly differentiate between the inside and the outside of the glomerulus. Moreover, because blocks are always within the glomerular area, gradient information in the same block between two glomeruli is expected to be more similar. Experiments revealed that the number of false positives was halved, keeping almost all true positives when using S-HOG compared to the R-HOG.

Related works

Segmentation is an important step to extract S-HOG descriptors. Recent studies on segmentation of the glomeruli are few [1, 2]. Nevertheless, there has been some research regarding segmentation of specific organs in general biomedical images, including region growing [15], level set method [16], and active contour model [17–19]. Most of these are semi-automatic, require the users' intervention, possess no guarantee of optimality [17–19], and are highly dependent on the initial solution provided by users as input. On the other hand, the segmentation algorithm developed in this study is ensured, theoretically, to obtain the optimal solution, producing high quality segmentation. In addition to the above-mentioned methods, more recent attempts include using deep learning [20]. Deep learning typically requires great computational and time resources, whereas the proposed algorithm can work even on a standard personal computer or a laptop.

The algorithm developed by Kvarnström et al. [21] is relevant to the proposed segmentation technique. Their algorithm for cell contour recognition is based on a dynamic program, where they first estimated the cell centers and constructed a ray from the center to each m direction (Fig. 1b), where $m = 32$. Then, they computed the boundary likeliness at n points on each ray, where they set $n = 30$. Their algorithm finds a smooth contour by

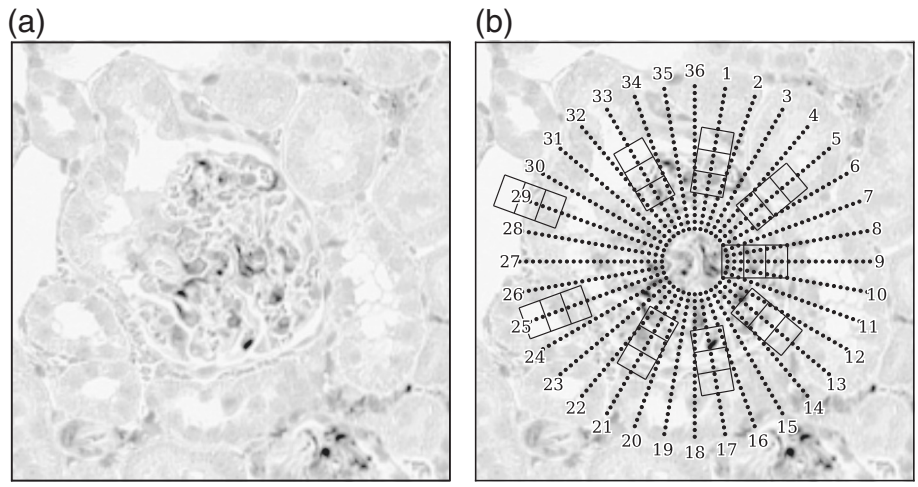

Fig. 1 Candidate glomeruli and line segments. Segmental HOG (S-HOG) is based on the boundary of the objects of interest. If the boundary of a candidate glomerulus (Panel (**a**)) is to be located, boundary likeliness is computed at every point on $m(=36)$ line segments placed uniformly in all m directions (Panel (**b**)). The boundary likeliness is computed at n points on each line segment. The n locations are depicted with dots in Panel (**b**)

taking a point on each ray to connect them. To this end, they posed the following discrete optimization problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} L_i(p_i), \qquad \text{wrt} \quad p_1,\ldots,p_m \in \{1,\ldots,n\}, \\ \text{subject to} \quad & |p_1 - p_2| \le \varsigma, \ldots, |p_{m-1} - p_m| \le \varsigma, |p_m - p_1| \le \varsigma, \end{aligned} \tag{1}$$

where $L_i : \{1,\ldots,n\} \to \mathbb{R}$ denotes the boundary likeliness function obtained by the sliding window technique, and $p_i \in \{1,\ldots,n\}$ $(i = 1,\ldots,m)$ is a location on the i-th line segment, where the line segment is discretized into n points numbered with a natural number. For instance, when $p_i = 1$, the i-th vertex is at the endpoint of the i-th line segment closest to the center, and the vertex can move from this endpoint to the other endpoint with increasing values of p_i. $L_i(p)$ is the boundary likeliness at the p-th location in the i-th line segment. The location p_i on the i-th line segment is more likely to be the boundary with a larger $L_i(p_i)$ value.

To obtain an optimal solution, Kvarnström et al. presented two algorithms. The first algorithm poses n sub-problems where in each sub-problem, the initial point, and the endpoint are the same. We shall refer to this algorithm here as the *exhaustive dynamic program* (EDP). Their second algorithm is a heuristic method that is faster than the first one, but possesses no guarantee for global optimality. In this study, we developed a new segmentation algorithm, referred to as *divide & conquer dynamic program* (DCDP). Compared to Kvarnström et al's algorithms [21], the DCDP algorithm has two advantages, as follows: not only is DCDP much faster than EDP, an exact optimal solution is always obtained; and the boundary likeliness function is trained with a machine learning technique to precisely estimate boundaries of the glomeruli.

One may consider another approach to the optimization problem (1), with a perspective that the problem is a formulation of finding a maximum a posteriori (MAP) estimation on a Markov random field (MRF) [22]. MRFs are a class of probabilistic models formulated on a graph. In the case of a problem (1), the graph has m nodes and forms a cycle. It is well known that the MAP estimation is efficient while using the Viterbi algorithm if the graph of the MRF is without cycles [23]. For graphs with cycles, many attempts such as LP relaxations [24] and max-product algorithms [25, 26] have been tried to compute approximate MAP estimations. Although these algorithms possess no guarantee to obtain an exact MAP estimation, they may perform well in practice. In this study, we empirically show that DCDP is much faster than these algorithms for glomeruli detection.

Contributions

The contributions of this study are summarized as follows:

- A new descriptor called S-HOG was developed to perform a comprehensive detection of hundreds of glomeruli in images of whole kidney sections. The new descriptor is equipped with flexible blocks that can be adaptively fitted to input images to acquire robustness for detection of glomeruli.
- In our experiments, the S-HOG descriptor halved the number of false positives, a limitation of the existing state-of-the-art descriptor R-HOG, while keeping almost all true positives.

- A new segmentation technique referred to as DCDP offered high-quality segmentation outputs that were used for the construction of the blocks in S-HOG. The worst computational time of the new algorithm is equal to the fastest existing segmentation algorithm, and our experimental results reveal that the new algorithm performs overwhelmingly faster in practice.

Methods

In this study, a new descriptor, S-HOG, has been proposed for the detection of the glomeruli in kidney microscopic images. Segmentation of the glomeruli is needed to extract the S-HOG descriptor. For rapid detection of the glomeruli in highly enlarged microscopic images, pre-screening is performed with R-HOG, which does not require prior segmentation. The proposed method consists of the following three stages (Fig. 2):

- the pre-screening stage,
- the segmentation stage, and
- the classification stage.

In each stage, a support vector machine (SVM) [27, 28] is used with a different type of HOG descriptor, resulting in three SVMs in total. To obtain the S-HOG descriptor, we performed segmentation of the glomeruli from the sub-images that passed the pre-screening (Fig. 2).

In what follows, we present the details of each stage, and then discuss how training datasets for each SVM are constructed and the materials used in the experiments. Finally, this section concludes with presenting a new segmentation algorithm, DCDP that is used for determining the blocks of S-HOG descriptors.

Pre-screening

In the pre-screening stage, candidate glomeruli are detected from a kidney microscopic image using the sliding window technique. The window size is set to 200×200 in our experiments. R-HOG features, which are 512-dimensional vectors based on our selected parameter values, are extracted and judged by SVM, and non-maximal suppression is then performed to obtain the candidate glomeruli. This stage is exactly the same as in the method developed in our previous studies [10, 11]. However, the subsequent two stages dramatically reduce the false positives detected by the method. Our experiment outputs, described in the 'Results and discussion' section, confirm that the non-maximal suppression successfully puts the center of the window in the glomerulus, which is crucial in the segmentation step.

Segmentation

Segmentation of the glomeruli is performed on sub-images that passed the pre-screening. In the segmentation algorithm, the boundary of a glomerulus is represented by an m-sided polygon whose m vertices are restricted to lie on m line segments, respectively. The m line segments are placed uniformly, as outlined by the dotted lines in Fig. 1b where $m = 36$. To determine the location of the vertex on each line segment, the sliding window technique is employed again.[1] The window sweeps through the line segment and computes the *boundary likeliness* at

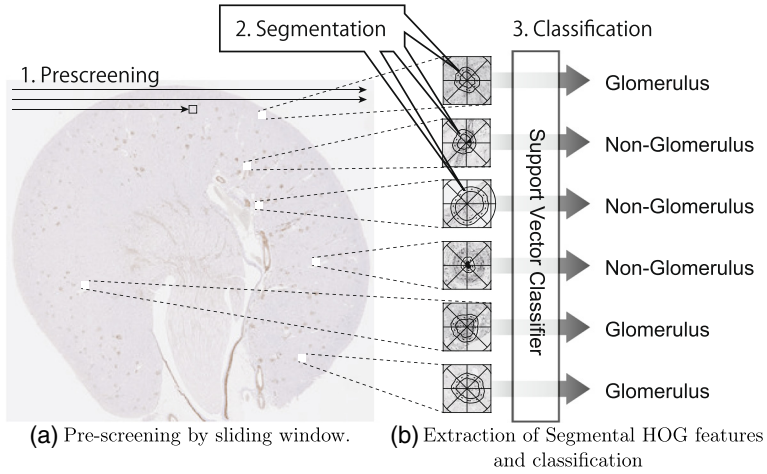

Fig. 2 Flow of our method. In this study, a new descriptor, Segmental HOG, was developed to detect glomeruli in highly magnified microscopic images. SVM is combined with Segmental HOG to classify candidate glomeruli that passed the pre-screening stage. To do this, our detection algorithm consists of three stages. (**a**) In the pre-screening stage, candidate glomeruli are searched in the entire microscopic image. (**b**) In the segmentation stage, the boundaries of each candidate glomerulus are estimated and Segmental HOG is based on the estimated boundaries. In the classification stage, feature vectors are extracted based on the estimated segmentations, and SVM is applied to judge whether each candidate glomeruli is positive or negative

n locations on the line segments. In Fig. 1b, the boundary likeliness L_i $(i = 1, \ldots, m)$ is computed at every dotted location. How L_i is computed is discussed at the end of this 'Segmentation' subsection. We set the length of the line segment to 63 pixels, where the endpoint closest to the center of the image is 17 pixels away from the center. The length between adjacent dots along a line segment is equal to 3 pixels, resulting to $n = 22$ dots on each line segment. In total, the boundary likeliness is computed at $mn(= 36 \times 22 = 792)$ locations. The values of the boundary likeliness are depicted by the green dots in Fig. 3a. Larger marks have higher boundary likeliness. To determine the vertices of the m-sided polygon, one might consider naïvely locating the points that achieve the highest boundary likeliness on each line segment. However, this approach often yields an extremely zigzag boundary (Fig. 3a).

To obtain a smoother boundary, Kvarnström et al. [21] imposed a constraint that suppresses distant adjacent vertices, and they established the maximization problem (1). Although our implementation of the boundary likeliness is different, the formulation of the problem to find an m-sided polygon is the same as (1), where in our experiments, ς is set to $\varsigma = 1$. Fig. 3 shows an example of the solution to the optimization problem. The m-sided polygon shown in Fig. 3b is the optimal solution to the

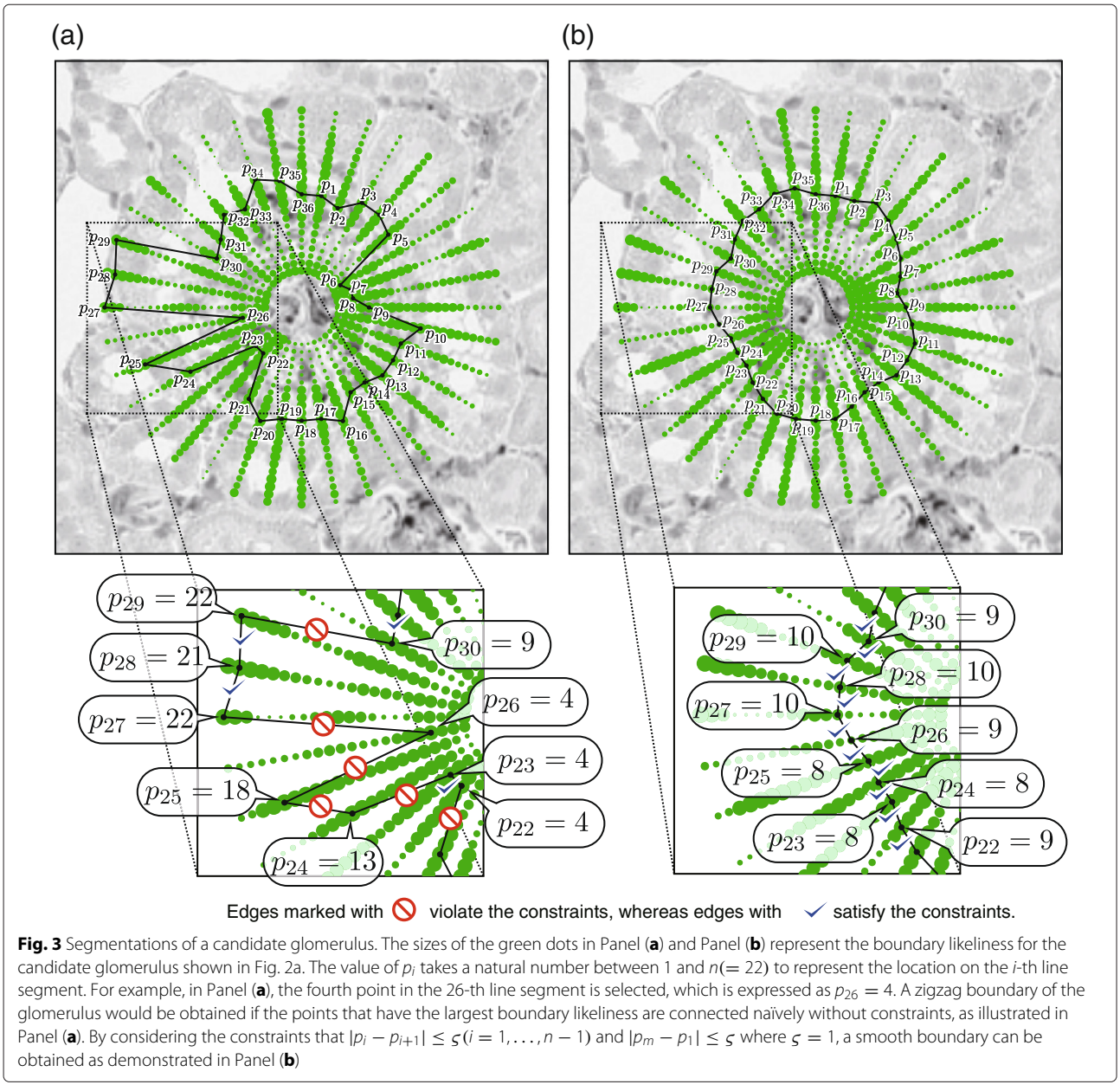

Fig. 3 Segmentations of a candidate glomerulus. The sizes of the green dots in Panel (**a**) and Panel (**b**) represent the boundary likeliness for the candidate glomerulus shown in Fig. 2a. The value of p_i takes a natural number between 1 and $n(= 22)$ to represent the location on the i-th line segment. For example, in Panel (**a**), the fourth point in the 26-th line segment is selected, which is expressed as $p_{26} = 4$. A zigzag boundary of the glomerulus would be obtained if the points that have the largest boundary likeliness are connected naïvely without constraints, as illustrated in Panel (**a**). By considering the constraints that $|p_i - p_{i+1}| \leq \varsigma (i = 1, \ldots, n-1)$ and $|p_m - p_1| \leq \varsigma$ where $\varsigma = 1$, a smooth boundary can be obtained as demonstrated in Panel (**b**)

maximization problem in (1). Compared to the solution without the constraint (Fig. 3a), it is apparent that the estimated boundary is formfitting to the true boundary by introducing the constraint. The details of the new algorithm for finding the optimal solution to (1) developed in this study is presented at the end of this section.

Computation of boundary likeliness $L_i(\cdot)$ The sliding window technique is employed in order to determine the vertices of the m-sided polygon described above. The window size in this stage is set to 30×15 pixels, and the windows sweep through the line segments (Fig. 1b). Each time the sliding window moves, a feature descriptor is computed from the window and is applied with linear SVM to compute the SVM score, which is what we refer to as the boundary likeliness $L_i(\cdot)$ (Fig. 3). The SVM scores of m vertices, $L_i(1), \ldots, L_i(n)$, are then obtained for $i = 1, \ldots, m$, and integrated in the maximization problem (1).

The HOG feature is adopted as the descriptor to compute the boundary likeliness. Each window is divided into three blocks, as shown in Fig. 1b. This division design is from an observation that some glomeruli are surrounded with a thick Bowman's capsule, and that the middle block is expected to capture this glomerular capsule. The statistics of nine discretely oriented gradients are computed in each block, producing a 27-dimensional feature vector.

Classification with the S-HOG descriptor

Candidate glomeruli obtained via pre-screening are classified using the proposed S-HOG descriptor. S-HOG exploits the glomerulus boundary located in the segmentation stage to generate 24 non-overlapping blocks, as shown in Fig. 4b.

Various types of glomeruli are observed on kidney microscopic images; some of them are surrounded by a thick Bowman's capsule. To effectively exploit this characteristic, the circle containing a candidate glomerulus is divided into the following three zones: the inner zone, middle zone, and outer zone. We divide the circle into eight disjoint sectors, and take the intersection of each zone and each sector to get 24 non-overlapping blocks (Fig. 4b), and gradients are then histogrammed for each block (Fig. 4d). In our experiments, we employed nine discretized oriented gradients, and SVM was applied to S-HOG feature vectors to discriminate between glomeruli and other regions.

Construction of training data

A total of three linear SVMs are used, one each for the pre-screening, segmentation, and classification stage, respectively. A training dataset is required for each of the three SVMs. Details on the construction of each training data set are given below.

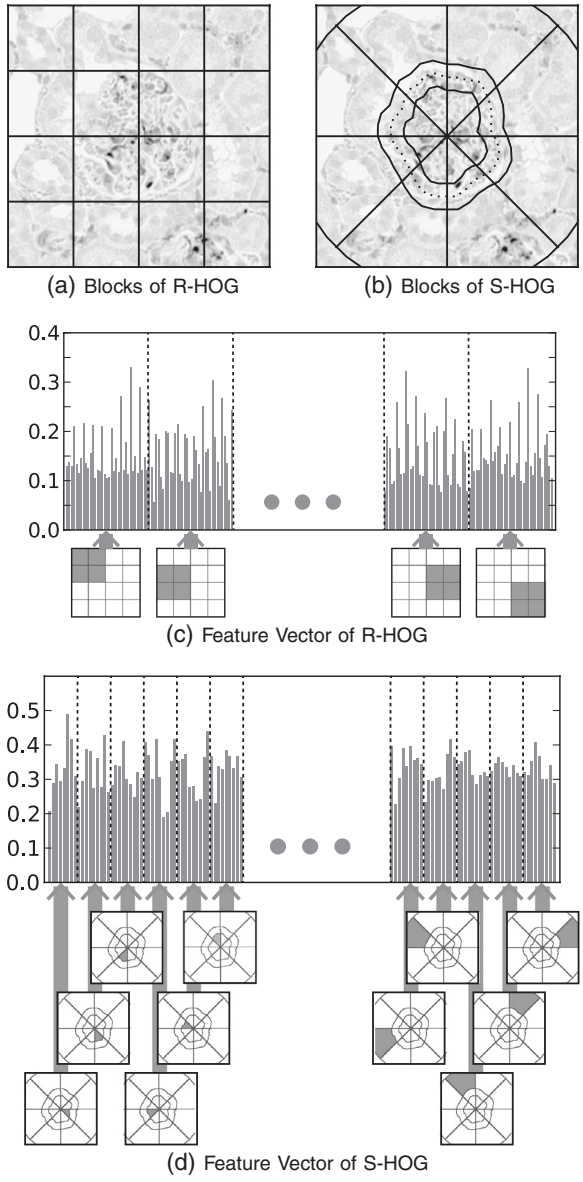

Fig. 4 Rectangular HOG and Segmental HOG. R-HOG has been used for object detection in many applications. R-HOG is the concatenation of statistics in each block in a grid that divides a rectangular region (**a**) and (**b**). On the other hand, the blocks of the proposed descriptor, S-HOG, are based on the segmentation of the object of interest (**b**) and (**d**)

Training data for the pre-screening stage Each example in the training data for pre-screening is a 200×200 sub-image. A positive example contains a glomerulus in the center of the sub-image, while a negative example does not. To gather these samples, the locations of the glomeruli within the whole-kidney-section images used for training are first annotated manually. Small glomeruli whose diameters are less than 50 pixels were ignored. Positive examples are the sub-images from 200×200

bounding boxes containing an annotated glomerulus in the center. Negative examples are 200×200 sub-images picked from random locations on the kidney microscopic images. From each sample, a 512-dimensional R-HOG descriptor is extracted.

Training data for the segmentation stage As described in the 'Segmentation' subsection, the boundary likeliness is computed in every position of the m line segment. This boundary likeliness is the SVM score. The position lying on the true boundary of a glomerulus is considered a positive example for the SVM, and the other positions are negative examples. To construct the training data for segmentation, the positive sub-images in the training dataset for pre-screening are reused.

Training data for the classification stage Examples used in the training data for pre-screening are used again for training in the classification stage, but with a different set of features extracted via S-HOG. For each training data sample, the previously described segmentation algorithm estimates the boundary of the glomerulus. Based on the estimated boundary, the statistics of oriented gradients are computed to obtain S-HOG feature vectors. This procedure is done for both positive and negative examples, even though negative examples do not contain a glomerulus.

Materials

The images used in the present study had been generated in a previous study [10], and only an overview is given in this subsection.

Twenty male 6-week-old SD and SDT rats were purchased from CLEA, Inc. (Tokyo, Japan) and were housed with a 12-h light-dark cycle and free access to water and chow.

At 16 or 24 weeks of age, SD and SDT rats (The number of rats is five for each group) were euthanized under ether anesthesia. Their kidneys were removed and immediately fixed in 10 % neutralized buffered formalin. The formalin-fixed kidneys were embedded in paraffin. For immunohistochemistry, kidney paraffin sections were deparaffinized and incubated overnight at 4 °C with anti-desmin mouse monoclonal antibody (Dako, Glostrup, Denmark) followed by horseradish peroxidase-conjugated secondary antibody (anti-mouse immunoglobulin goat polyclonal antibody; Nichirei, Tokyo, Japan). The sections were stained brown with 3,3'-diaminobenzidine. Whole slide images of the sections were obtained with Aperio Scan Scope XT (Leica Microsystems, Wetzlar, Germany). All animal experiments were performed in accordance with the Act on Welfare and Management of Animals and the institutional guidelines, and approved by the institutional Committee of Animal Experiments of New Drug Development Research Center Inc. (Hokkaido, Japan).

A total of 20 whole-kidney-section images were used in our experiments. The image sizes were $9,849 \times 10,944$ pixels in average. Each image was from one of four groups: 16-week-old SD rats, 16-week-old SDT rats, 24-week-old SD rats, and 24-week-old SDT rats. Henceforth, for simplicity, we will refer to them as 16SD, 16SDT, 24SD, 24SDT, respectively, each group containing five images. For performance evaluation, we manually annotated every glomerulus in the images. We divided the image set into five subsets: Set A, Set B, Set C, Set D, and Set E. Each subset consists of a 16SD image, a 16SDT image, a 24SD image, and a 24SDT image. For assessment of detection performance, the position of every glomerulus in the images is annotated, and for evaluation of segmentation performance, the areas of the glomeruli in Set A and Set B are located manually using a graphics software.

Divide & conquer dynamic program (DCDP)

Herein, a new algorithm named Divide & Conquer Dynamic Program (DCDP) for solving the optimization problem (1) is presented. The new algorithm DCDP also takes $O(n^2 m\varsigma)$ computational time in worst-case scenarios, although the new algorithm solves the same problem much faster than EDP, as presented in the 'Results and discussion' section.

Let us denote the objective function by $J(\boldsymbol{p})$, i.e., $J(\boldsymbol{p}) := \sum_{i=1}^{m} L_i(p_i)$, and observe that the problem (1) can be solved in $O(nm\varsigma)$ computational time by a dynamic program if one of the constraints $|p_m - p_1| \leq \varsigma$ is disregarded. The idea to devise the new algorithm is based on the following fact: Suppose $\boldsymbol{p}_0^\star$ is an optimal solution that maximizes $J(\cdot)$ without the constraint $|p_m - p_1| \leq \varsigma$. Then if $\boldsymbol{p}_0^\star$ is a feasible solution for the original problem (1), it is also an optimal solution.

To express this idea mathematically, let us define

$$\begin{aligned}\mathcal{S}(\mathcal{I}) := \big\{\boldsymbol{p} = (p_1, \ldots, p_m) \in \mathbb{N}_n^m \mid \\ p_m \in \mathcal{I},\ |p_1 - p_2| \leq \varsigma, \ldots, |p_{m-1} - p_m| \leq \varsigma, \\ |p_m - p_1| \leq \varsigma\big\}.\end{aligned}$$

for $\mathcal{I} \subseteq \mathbb{N}_n$, where $\mathbb{N}_n := \{1, \ldots, n\}$. Note that $\mathcal{S}(\mathbb{N}_n)$ is the feasible region of the original problem (1). The goal of DCDP is to find an optimal solution

$$\boldsymbol{p}^\star \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathbb{N}_n)} J(\boldsymbol{p}).$$

Dynamic program (DP) cannot solve this problem directly owing to the existence of the constraint $|p_m - p_1| \leq \varsigma$. To use DP, we consider finding the maximizer of $J(\boldsymbol{p})$ from a relaxed region $\mathcal{S}_{\mathrm{L}}(\mathbb{N}_n)$, where $\mathcal{S}_{\mathrm{L}}(\mathcal{I})$ is defined as

$$\mathcal{S}_{\mathrm{L}}(\mathcal{I}) := \big\{\boldsymbol{p} = (p_1, \ldots, p_m) \in \mathbb{N}_n^m \mid p_m \in \mathcal{I},\, p_1 \in \mathcal{I} + \{-\varsigma, \ldots, +\varsigma\},\, |p_1 - p_2| \le \varsigma, \ldots, |p_{m-1} - p_m| \le \varsigma \big\},$$

where the operator $+$ denotes that for any two sets $\mathcal{I}$ and $\mathcal{J}$, $\mathcal{I}+\mathcal{J} := \{i+j \mid i \in \mathcal{I},\, j \in \mathcal{J}\}$. Note that $\mathcal{S}(\mathcal{I}) \subseteq \mathcal{S}_{\mathrm{L}}(\mathcal{I})$. The strategy of DCDP is to first find the solution of the relaxed problem,

$$\boldsymbol{p}_0^\star \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathbb{N}_n)} J(\boldsymbol{p})$$

and then check the feasibility: if $\boldsymbol{p}_0^\star \in \mathcal{S}(\mathbb{N}_n)$, then $\boldsymbol{p}_0^\star$ is the optimal solution of the original problem (1). If $\boldsymbol{p}_0^\star \notin \mathcal{S}(\mathbb{N}_n)$, then the set $\mathbb{N}_n$ is divided into $\mathcal{I}_1$ and $\mathcal{I}_2$ (i.e. $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathbb{N}_n$), and the following two sub-problems are solved:

$$\boldsymbol{p}_1^\star \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p}) \qquad \text{and} \qquad \boldsymbol{p}_2^\star \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_2)} J(\boldsymbol{p}).$$

Notice that the original feasible region, $\mathcal{S}(\mathbb{N}_n)$, is the sum of the two regions, $\mathcal{S}(\mathcal{I}_1)$ and $\mathcal{S}(\mathcal{I}_2)$. Therefore, we can take either of the two solutions, $\boldsymbol{p}_1^\star$ and $\boldsymbol{p}_2^\star$, whichever has the larger objective value. DCDP employs a divide and conquer approach that repeatedly applies the above strategy to sub-problems. The basic approach of DCDP is summarized in Algorithm 1. Invoking the function DCDP_Basic($\mathbb{N}_n$) yields the optimal solution for the original problem. Here, the function $(\mathcal{I}_1, \mathcal{I}_2) := \mathrm{Split}(\mathcal{I}_0)$ divides the set $\mathcal{I}_0$ into two exclusive non-empty subsets, $\mathcal{I}_1$ and $\mathcal{I}_2$.

Algorithm 1 $\boldsymbol{p}_0 = \mathrm{DCDP\_Basic}(\mathcal{I}_0)$

Input: $\mathcal{I}_0 \subseteq \mathbb{N}_n$.
Output: $\boldsymbol{p}_0 \in \operatorname{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$.

$\boldsymbol{p}_0 \in \operatorname{argmax}_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$;
if $\boldsymbol{p}_0 \in \mathcal{S}(\mathcal{I}_0)$, **then return**;
$(\mathcal{I}_1, \mathcal{I}_2) := \mathrm{Split}(\mathcal{I}_0)$;
$\boldsymbol{p}_1 := \mathrm{DCDP\_Basic}(\mathcal{I}_1)$;
$\boldsymbol{p}_2 := \mathrm{DCDP\_Basic}(\mathcal{I}_2)$;
$i^\star \in \operatorname{argmax}_{i \in \{1,2\}} J(\boldsymbol{p}_i)$;
$\boldsymbol{p}_0 := \boldsymbol{p}_{i^\star}$;

The first step $\boldsymbol{p}_0 \in \operatorname{argmax}_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$ can be performed in $O(nm\varsigma)$ computational time. An instance of the dynamic program is given in Algorithm 2. Note that $\boldsymbol{p}_0 \in \mathcal{S}(\mathcal{I}_0)$ is always ensured if the cardinality of $\mathcal{I}_0$ is one, because the relaxed region is reduced to the unrelaxed region (i.e. $\mathcal{S}(\{h\}) = \mathcal{S}_{\mathrm{L}}(\{h\})$). The function DCDP_Basic is invoked, at most, $(2n-1)$ times. This implies that the computational time in worst cases is $O(n^2 m\varsigma)$. As will be shown in the 'Results and discussion' section, we empirically found that the number of invoking the function recursively is much smaller than $(2n-1)$.

Algorithm 2 $O(nm\varsigma)$ Dynamic program for $\max_{\boldsymbol{p}^\star \in \mathcal{S}_{\mathrm{L}}(\mathcal{I})} J(\boldsymbol{p})$

Input: $\mathcal{I} \subseteq \mathbb{N}_n$.
Output: $\boldsymbol{p} \in \operatorname{argmax}_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I})} J(\boldsymbol{p})$

Initialize all entries in the $n \times m$ matrix Q with $-\infty$.
for $j \in \mathcal{I} + [-\varsigma, +\varsigma] \cap \mathbb{N}_n$ **do**
  $Q(j, 1) := L_1(j)$;
end for
for $t = 2, \ldots, m$ **do**
  if $t < m$, **then** $\mathcal{I}_t := \mathbb{N}_n$ else $\mathcal{I}_t := \mathcal{I}$;
  for $i \in \mathcal{I}_t$ **do**
    $j_\star := \operatorname{argmax}\{Q(j, t-1) \mid j \in [\, i - \varsigma, i + \varsigma] \cap \mathbb{N}_n\}$;
    $Q(i, t) := L_t(i) + Q(j_\star, t-1)$; $P(i, t) := j_\star$;
  end for
end for
$p_m^\star \in \operatorname{argmax}_{i \in \mathcal{I}} Q(i, m)$;
$t := m - 1$;
while $t \ge 1$ **do**
  $p_t^\star := P(p_{t+1}^\star, t+1)$; $t := t - 1$;
end while

In the text below, the pruning steps and the resulting accelerated DCDP algorithm are detailed. Mathematical proof that the DCDP algorithm is guaranteed to obtain an optimal solution is also given. This property is favorable compared to MAP estimation methods-such as LP relaxation-that does not always achieve an optimal solution. For the implementation of Split($\mathcal{I}_0$), we considered the following three schemes: *Half Split*, *Max Split*, and *Adap Split*. These splitting schemes are described at the end of this section.

Pruning

Pruning can accelerate the DCDP algorithm. Consider the case where the lower boundary is ℓ, such that

$$\max_{\boldsymbol{p} \in \mathcal{S}(\mathbb{N}_n)} J(\boldsymbol{p}) \ge \ell,$$

is known in advance when searching for the solution in $\mathcal{S}(\mathcal{I}_0)$. If $\ell > \max_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$, then no optimal solution is in $\mathcal{S}(\mathcal{I}_0)$ because

$$\max_{\boldsymbol{p} \in \mathcal{S}(\mathbb{N}_n)} J(\boldsymbol{p}) \ge \ell > \max_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p}) \ge \max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p}).$$

Based on this fact, the pruning step is added to obtain Algorithm 3, and we have the following lemma:

Lemma 1. For any subset $\mathcal{I}_0 \subseteq \mathbb{N}_n$ and $\forall \ell \in \mathbb{R} \cup \{-\infty\}$, when the algorithm runs with $(\boldsymbol{p}_0, J_0, \ell_0) = \mathrm{DCDP}(\mathcal{I}_0, \ell)$, the returned tuple $(\boldsymbol{p}_0, J_0, \ell_0)$ satisfies one of the following:

- **Case G:** If $\max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p}) \ge \ell$, then
$$\boldsymbol{p}_0 \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p}), \quad \text{and} \quad J_0 = \ell_0 = J(\boldsymbol{p}_0) \ge \ell.$$
- **Case L:** If $\max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p}) < \ell$, then $J_0 < \ell_0 = \ell$.

Algorithm 3 $(\boldsymbol{p}_0, J_0, \ell_0) = \mathrm{DCDP}(\mathcal{I}_0, \ell)$. Modification of DCDP_Basic by adding pruning steps

Input: $\mathcal{I}_0 \subseteq \mathbb{N}_n$.

$\boldsymbol{p}_{\mathrm{L}} \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$;
if $\ell > J(\boldsymbol{p}_{\mathrm{L}})$ **then**
  {Rule A}
  $J_0 := -\infty$; $\ell_0 := \ell$;
  return;
end if
if $\boldsymbol{p}_{\mathrm{L}} \in \mathcal{S}(\mathcal{I}_0)$ **then**
  {Rule B}
  $\boldsymbol{p}_0 := \boldsymbol{p}_{\mathrm{L}}$; $J_0 := J(\boldsymbol{p}_{\mathrm{L}})$; $\ell_0 := \max(\ell, J_0)$);
  return;
end if
{Rule C}
$(\mathcal{I}_1, \mathcal{I}_2) := \mathrm{Split}(\mathcal{I}_0)$;
$(\boldsymbol{p}_1, J_1, \ell_1) := \mathrm{DCDP}(\mathcal{I}_1, \ell)$;
$(\boldsymbol{p}_2, J_2, \ell_2) := \mathrm{DCDP}(\mathcal{I}_2, \ell_1)$;
$i^\star \in \mathrm{argmax}_{i \in \{1,2\}} J_i$;
$\boldsymbol{p}_0 := \boldsymbol{p}_{i^\star}$; $J_0 := J(\boldsymbol{p}_{i^\star})$;
$\ell_0 := \max(\ell_2, J_0)$;

Proof of Lemma 1

We shall use the following notation: For any $\mathcal{I} \subseteq \mathbb{N}_n$,

$$J(\mathcal{I}) := \max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I})} J(\boldsymbol{p}), \quad \text{and} \quad J_{\mathrm{L}}(\mathcal{I}) := \max_{\boldsymbol{p} \in \mathcal{S}_{\mathrm{L}}(\mathcal{I})} J(\boldsymbol{p}).$$

The following relationships will be used in this proof:

$$J(\boldsymbol{p}_{\mathrm{L}}) \overset{\text{(eqA)}}{=} J_{\mathrm{L}}(\mathcal{I}_0) \overset{\text{(ineqA)}}{\geq} J(\mathcal{I}_0) \overset{\text{(eqB)}}{=} \max(J(\mathcal{I}_1), J(\mathcal{I}_2)),$$

where the labels (eqA), (ineqA), and (eqB) are used to distinguish these equalities and the inequality in later descriptions of this proof. The inequality follows from the fact that $\mathcal{S}(\mathcal{I}_0) \subseteq \mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)$, while the second equality eqB follows from $\mathcal{S}(\mathcal{I}_0) = \mathcal{S}(\mathcal{I}_1) \cup \mathcal{S}(\mathcal{I}_2)$.

We will prove the lemma by induction. For the case where $\mathrm{card}(\mathcal{I}_0) = 1$, observe that $\mathcal{S}_{\mathrm{L}}(\mathcal{I}_0) = \mathcal{S}(\mathcal{I}_0)$, implying that

$$\boldsymbol{p}_{\mathrm{L}} \in \operatorname*{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p}) \quad \text{and} \quad J(\mathcal{I}_0) = J_{\mathrm{L}}(\mathcal{I}_0) = J_0.$$

If $J(\mathcal{I}_0) = J(\boldsymbol{p}_{\mathrm{L}}) < \ell$, then by Rule A in the algorithm, $J_0 = -\infty$ and $J_0 < \ell_0 = \ell$. On the other hand, if $J(\boldsymbol{p}_{\mathrm{L}}) \geq \ell$, then as $\boldsymbol{p}_{\mathrm{L}} \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$, we have $\boldsymbol{p}_{\mathrm{L}} \in \mathcal{S}(\mathcal{I}_0)$. Thus, by Rule B, we have $\boldsymbol{p}_0 = \boldsymbol{p}_{\mathrm{L}}$ and $J_0 = \ell_0 \geq \ell$. Therefore, the lemma is true for $\mathrm{card}(\mathcal{I}_0) = 1$.

Let us now assume that the lemma holds for any $\mathcal{I}_0 \subseteq \mathbb{N}_n$, such that $\mathrm{card}(\mathcal{I}_0) < k$, to show that the lemma is also established for any $\mathcal{I}_0$, such that $\mathrm{card}(\mathcal{I}_0) = k$. Now suppose that $\mathcal{I}_0 \subseteq \mathbb{N}_n$ and $\mathrm{card}(\mathcal{I}_0) = k$. The following is an exhaustive list of all possible cases:

(1) $\ell > J_{\mathrm{L}}(\mathcal{I}_0)$,
(2) $J_{\mathrm{L}}(\mathcal{I}_0) \geq \ell$ and $\boldsymbol{p}_{\mathrm{L}} \in \mathcal{S}(\mathcal{I}_0)$,
(3) $J_{\mathrm{L}}(\mathcal{I}_0) \geq J(\mathcal{I}_0) = J(\mathcal{I}_1) = J(\mathcal{I}_2) \geq \ell$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$,
(4) $J_{\mathrm{L}}(\mathcal{I}_0) \geq J(\mathcal{I}_0) = J(\mathcal{I}_1) \geq \ell > J(\mathcal{I}_2)$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$,
(5) $J_{\mathrm{L}}(\mathcal{I}_0) \geq J(\mathcal{I}_0) = J(\mathcal{I}_1) > J(\mathcal{I}_2) \geq \ell$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$,
(6) $J_{\mathrm{L}}(\mathcal{I}_0) \geq J(\mathcal{I}_0) = J(\mathcal{I}_2) > J(\mathcal{I}_1) \geq \ell$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$,
(7) $J_{\mathrm{L}}(\mathcal{I}_0) \geq J(\mathcal{I}_0) = J(\mathcal{I}_2) \geq \ell > J(\mathcal{I}_1)$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$,
(8) $J_{\mathrm{L}}(\mathcal{I}_0) \geq \ell > J(\mathcal{I}_0)$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$.

We shall show that for each of the seven cases above, either Case G or Case L is true.

(1) As $\ell > J_{\mathrm{L}}(\mathcal{I}_0) \overset{\text{(eqA)}}{=} J(\boldsymbol{p}_{\mathrm{L}}) \overset{\text{(ineqA)}}{\geq} J(\mathcal{I}_0)$, then by Rule A, we have $J_0 = -\infty$ and $J_0 < \ell_0 = \ell$. Thus, Case L is satisfied.
(2) If $J_{\mathrm{L}}(\mathcal{I}_0) \geq \ell$, then $J(\boldsymbol{p}_{\mathrm{L}}) \geq \ell$. Moreover, $\boldsymbol{p}_{\mathrm{L}} \in \mathcal{S}(\mathcal{I}_0)$ implies $\boldsymbol{p}_{\mathrm{L}} \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$ and $J(\boldsymbol{p}_{\mathrm{L}}) = J(\mathcal{I}_0)$. Then by Rule B, $J_0 = J(\boldsymbol{p}_{\mathrm{L}}) = J(\mathcal{I}_0) = \ell_0$ and $\boldsymbol{p}_0 = \boldsymbol{p}_{\mathrm{L}} \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$. Therefore, Case G holds.
(3) Given $J(\boldsymbol{p}_{\mathrm{L}}) \overset{\text{(eqA)}}{=} J_{\mathrm{L}}(\mathcal{I}_0) \geq \ell$ and $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$, Rule C is applied. Observe that $(\boldsymbol{p}_1, J_1, \ell_1)$ satisfies Case G because $J(\mathcal{I}_1) = \max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p}) \geq \ell$. Thus, $J_1 = \ell_1 = J(\mathcal{I}_1) \geq \ell$. Similarly, $(\boldsymbol{p}_2, J_2, \ell_2)$ also satisfies Case G, and we have $J_2 = \ell_2 = J(\mathcal{I}_2) \overset{\text{(eqB)}}{=} J(\mathcal{I}_1) = J(\mathcal{I}_0)$. Therefore, the returned value $\boldsymbol{p}_0$ of $\mathrm{DCDP}(\mathcal{I}_0, \ell)$ is equal to $\boldsymbol{p}_1$ or $\boldsymbol{p}_2$, both of which are in $\mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$. Furthermore, we obtain $\ell_0 = J_0 = J_1 = J_2 = J(\mathcal{I}_0) \geq \ell$. Hence, we have Case G.
(4) As with Case (3), $(\boldsymbol{p}_1, J_1, \ell_1)$ satisfies Case G, and we have $\boldsymbol{p}_1 \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p})$ and $J_1 = \ell_1 = J(\mathcal{I}_1) \geq \ell$. Hence, it follows that $\max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_2)} J(\boldsymbol{p}_2) = J(\mathcal{I}_2) < \ell \leq \ell_1$, and Case L holds for $(\boldsymbol{p}_2, J_2, \ell_2)$ with $J_2 < \ell_2 = \ell_1$. Therefore, output $\boldsymbol{p}_0 = \boldsymbol{p}_1$ is in $\mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$ because $\mathcal{S}(\mathcal{I}_1) \subset \mathcal{S}(\mathcal{I}_0)$, and $J_0 = J_1 = \ell_2 = \ell_0 \geq \ell$. This gives us Case G.
(5) In a similar logic as in Case (4), Case G is true for $(\boldsymbol{p}_1, J_1, \ell_1)$, $\boldsymbol{p}_1 \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p})$, and $J_1 = \ell_1 = J(\mathcal{I}_1) \geq \ell$. Given that $J(\mathcal{I}_2) < J(\mathcal{I}_1)$, then $J(\mathcal{I}_2) < \ell_1$; hence, Case L also holds for $(\boldsymbol{p}_2, J_2, \ell_2)$ in this sub-case. Therefore, the same conclusion from Case (4) follows.
(6) As $\boldsymbol{p}_{\mathrm{L}} \notin \mathcal{S}(\mathcal{I}_0)$, Rule C is implemented. Note that $\max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p}) = J(\mathcal{I}_1) \geq \ell$, $(\boldsymbol{p}_1, J_1, \ell_1)$ follows Case G, and we have $J_1 = \ell_1 = J(\mathcal{I}_1) \geq \ell$. This implies that $J(\mathcal{I}_2) > J(\mathcal{I}_1) = \ell_1$. Therefore, Case G also holds for $(\boldsymbol{p}_2, J_2, \ell_2)$, and we obtain $J_2 = \ell_2 = J(\mathcal{I}_2) > J(\mathcal{I}_1) = J_1$. Thus, $\mathrm{DCDP}(\mathcal{I}_0, \ell)$ outputs $\boldsymbol{p}_0 = \boldsymbol{p}_2 \in \mathrm{argmax}_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$ and $J_0 = J_2 = \ell_2 = \ell_0 > \ell$. Hence, Case G holds.
(7) Similar to the previous cases, we apply Rule C. Given $\ell > J(\mathcal{I}_1) = \max_{\boldsymbol{p} \in \mathcal{S}(\mathcal{I}_1)} J(\boldsymbol{p})$, by Case L, $J_1 < \ell_1 = \ell$. and so it follows that $J(\mathcal{I}_2) \geq \ell_1 = \ell$. Therefore, $(\boldsymbol{p}_2, J_2, \ell_2)$ satisfies Case G, with $J_2 = \ell_2 = J(\mathcal{I}_2) > \ell$. Hence, the returned values of $\mathrm{DCDP}(\mathcal{I}_0, \ell)$ are

$\boldsymbol{p}_0 = \boldsymbol{p}_2 \in \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}(\mathcal{I}_0)} J(\boldsymbol{p})$ and $J_0 = J_2 = \ell_2 = \ell_0 > \ell$, and Case G is satisfied.

(8) Likewise, we implement Rule C for this case. Observe that because $\max(J(\mathcal{I}_1), J(\mathcal{I}_2)) \overset{(\text{eqB})}{=} J(\mathcal{I}_0) < \ell$, then both $(\boldsymbol{p}_1, J_1, \ell_1)$ and $(\boldsymbol{p}_2, J_2, \ell_2)$ satisfy Case L. Thus, we have $J_1 < \ell_1 = \ell$ and $J_2 < \ell_1 = \ell_2$. Therefore, DCDP$(\mathcal{I}_0, \ell)$ outputs $J_0 = \max(J_1, J_2) < \ell_2 = \ell_0 = \ell_1 = \ell$, which corresponds to Case L.

□

Finally, from Lemma 1, we can derive the following theorem, an important theoretical result of this study:

Theorem 1. The optimal solution of the problem (1) is obtained by invoking $(\boldsymbol{p}^\star, J^\star, \ell^\star) = \text{DCDP}(\mathbb{N}_n, -\infty)$.

Splitting schemes

For the implementation of Split$(\mathcal{I}_0)$, we considered three schemes: *Half Split*, *Max Split*, and *Adap Split*.

Half split In this scheme, the subset of indices $\mathcal{I}_0$ is simply divided into the first half and the second half. The resulting $\mathcal{I}_1$ and $\mathcal{I}_2$ are sets of consecutive integers. For instance, this scheme divides $\mathcal{I}_0 = \{7, 8, 9, 10, 11, 12\}$ into $\mathcal{I}_1 = \{7, 8, 9\}$ and $\mathcal{I}_2 = \{10, 11, 12\}$. To increase the lower-bound ℓ defined earlier, a heuristic process swaps $\mathcal{I}_1$ with $\mathcal{I}_2$ if

$$\max_{h\in\mathcal{I}_1} L_m(h) < \max_{h\in\mathcal{I}_2} L_m(h)$$

is employed.

Max split Similar to Half Split, $\mathcal{I}_1$ and $\mathcal{I}_2$ generated by Max Split are sets of consecutive integers, although the splitting points are different. The splitting point of Half Split is the center of the interval $\mathcal{I}_0$, whereas the splitting point of Max Split is given by $h_\star(\mathcal{I}_0) := \operatorname{argmax}_{h\in\mathcal{I}_0} L_m(h)$. For example, if $h_\star(\mathcal{I}_0) = 8$ for $\mathcal{I}_0 = \{7, 8, 9, 10, 11, 12\}$, this scheme outputs $\mathcal{I}_1 = \{7, 8\}$ and $\mathcal{I}_2 = \{9, 10, 11, 12\}$. In general, the resulting divisions are given by

$$\mathcal{I}_1 := \{h \in \mathcal{I}_0 \,|\, h \le h_\star(\mathcal{I}_0)\}, \quad \text{and} \quad \mathcal{I}_2 := \{h \in \mathcal{I}_0 \,|\, h > h_\star(\mathcal{I}_0)\}.$$

If $\text{card}(\mathcal{I}_2) = 0$, the entry $h_\star(\mathcal{I}_0)$ is moved from $\mathcal{I}_1$ to $\mathcal{I}_2$. A heuristic process swaps $\mathcal{I}_1$ with $\mathcal{I}_2$ if $\text{card}(\mathcal{I}_1) > \text{card}(\mathcal{I}_2)$ is applied.

Adap split Unlike the previous two schemes, Adap Split determines the splitting point adaptively using the current solution $\boldsymbol{p}_{\text{L},0} := \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\text{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$. Let us denote the first entry and the last entry in the vector $\boldsymbol{p}_{\text{L},0}$ as p_1^0 and p_m^0, respectively. Using Adap Split, the mean of p_1^0 and p_m^0 is set as the splitting point. For instance, if $p_1^0 = 9, p_m^0 = 12$ and $\mathcal{I}_0 = \{7, 8, 9, 10, 11, 12\}$, then this scheme divides $\mathcal{I}_0$ into $\mathcal{I}_1 = \{7, 8, 9, 10\}$ and $\mathcal{I}_2 = \{11, 12\}$. In general, the resulting divisions are given by

$$\mathcal{I}_1 := \left\{ h \in \mathcal{I}_0 \,|\, h \le \frac{p_1^0 + p_m^0}{2} \right\}, \quad \text{and}$$

$$\mathcal{I}_2 := \left\{ h \in \mathcal{I}_0 \,|\, h > \frac{p_1^0 + p_m^0}{2} \right\}.$$

The smallest entry in $\mathcal{I}_2$ is moved to $\mathcal{I}_1$ if $\text{card}(\mathcal{I}_1) = 0$, and the largest entry in $\mathcal{I}_1$ is moved to $\mathcal{I}_2$ if $\text{card}(\mathcal{I}_2) = 0$. A swapping heuristic process similar to that used in Max Split is then applied.

Why Adap split is better Adap Split is expected to be the smartest heuristic process among the three splitting schemes. To support this claim, we illustrate the process of DCDP on a small toy problem with $(n, m, \varsigma) = (8, 12, 1)$, as shown in Fig. 5. The original problem and the relaxed problem are depicted in Fig. 5a and b, respectively. When running DCDP$(\mathbb{N}_8, -\infty)$, where $\mathcal{I}_0 = \mathbb{N}_8$, it was observed that $\boldsymbol{p}_{\text{L},0} := \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\text{L}}(\mathcal{I}_0)} J(\boldsymbol{p}) \notin \mathcal{S}(\mathcal{I}_0)$. Thereby the set $\mathcal{I}_0$ is divided into $\mathcal{I}_1$ and $\mathcal{I}_2$ to produce two new branches DCDP$(\mathcal{I}_1, -\infty)$ and DCDP$(\mathcal{I}_2, \ell_1)$ where ℓ_1 will be computed via DCDP$(\mathcal{I}_1, -\infty)$.

If Adap Split is employed, the two subsets are $\mathcal{I}_1 = \{7, 8\}$ and $\mathcal{I}_2 = \{1, 2, 3, 4, 5, 6\}$. In DCDP$(\mathcal{I}_1, -\infty)$, $\boldsymbol{p}_{\text{L},1} := \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\text{L}}(\mathcal{I}_1)} J(\boldsymbol{p})$ is in $\mathcal{S}(\mathcal{I}_1)$ (as shown in Fig. 5c), implying that $\boldsymbol{p}_{\text{L},1}$ is the maximizer of $J(\boldsymbol{p})$ over $\mathcal{S}(\mathcal{I}_1)$ and no more branching occurs. The value of ℓ_1 is computed and we obtain $\ell_1 = J(\boldsymbol{p}_{\text{L},1}) = 10.1$. Next, DCDP$(\mathcal{I}_2, \ell_1)$ is invoked and $\boldsymbol{p}_{\text{L},2} := \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\text{L}}(\mathcal{I}_2)} J(\boldsymbol{p}) = 9.5$ is obtained (Fig. 5d). However, $J(\boldsymbol{p}_{\text{L},2}) = 9.5 < 10.1 = \ell_1$, which implies that the optimal solution is not in $\mathcal{S}(\mathcal{I}_2)$. Hence, $\boldsymbol{p}_{\text{L},1}$ is the optimal solution of the original problem.

On the other hand, if Half Split is applied, the set $\mathcal{I}_0 = \mathbb{N}_8$ is divided into $\mathcal{I}_1 = \{5, 6, 7, 8\}$ and $\mathcal{I}_2 = \{1, 2, 3, 4\}$ (Fig. 5e and f). In DCDP$(\mathcal{I}_1, -\infty)$, the obtained solution of the relaxed problem is $\boldsymbol{p}_{\text{L},1} := \operatorname{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\text{L}}(\mathcal{I}_1)} J(\boldsymbol{p}) = \boldsymbol{p}_{\text{L},0}$, which is, again, not in $\mathcal{S}(\mathcal{I}_1)$, leading to further branching along this sub-problem (Fig. 5e). In fact, in our experiments discussed in the 'Results and discussion' section, it was observed that both Half Split and Max Split frequently encounter cases where $\boldsymbol{p}_{\text{L},1} = \boldsymbol{p}_{\text{L},0}$ or $\boldsymbol{p}_{\text{L},2} = \boldsymbol{p}_{\text{L},0}$. Whenever $\boldsymbol{p}_{\text{L},1} = \boldsymbol{p}_{\text{L},0}$, additional new branches for the divisions of $\mathcal{I}_1$ are produced because $\boldsymbol{p}_{\text{L},1} = \boldsymbol{p}_{\text{L},0} \notin \mathcal{S}(\mathcal{I}_0)$, leading to $\boldsymbol{p}_{\text{L},1} \notin \mathcal{S}(\mathcal{I}_1) \subset \mathcal{S}(\mathcal{I}_0)$. Similarly, new branches for the divisions of $\mathcal{I}_2$ are also generated when $\boldsymbol{p}_{\text{L},2} = \boldsymbol{p}_{\text{L},0}$.

In brief, the Adap Split performs better because $\boldsymbol{p}_{\text{L},1} = \boldsymbol{p}_{\text{L},0}$ or $\boldsymbol{p}_{\text{L},2} = \boldsymbol{p}_{\text{L},0}$ is less likely to happen in this scheme, thus resulting in less branches.

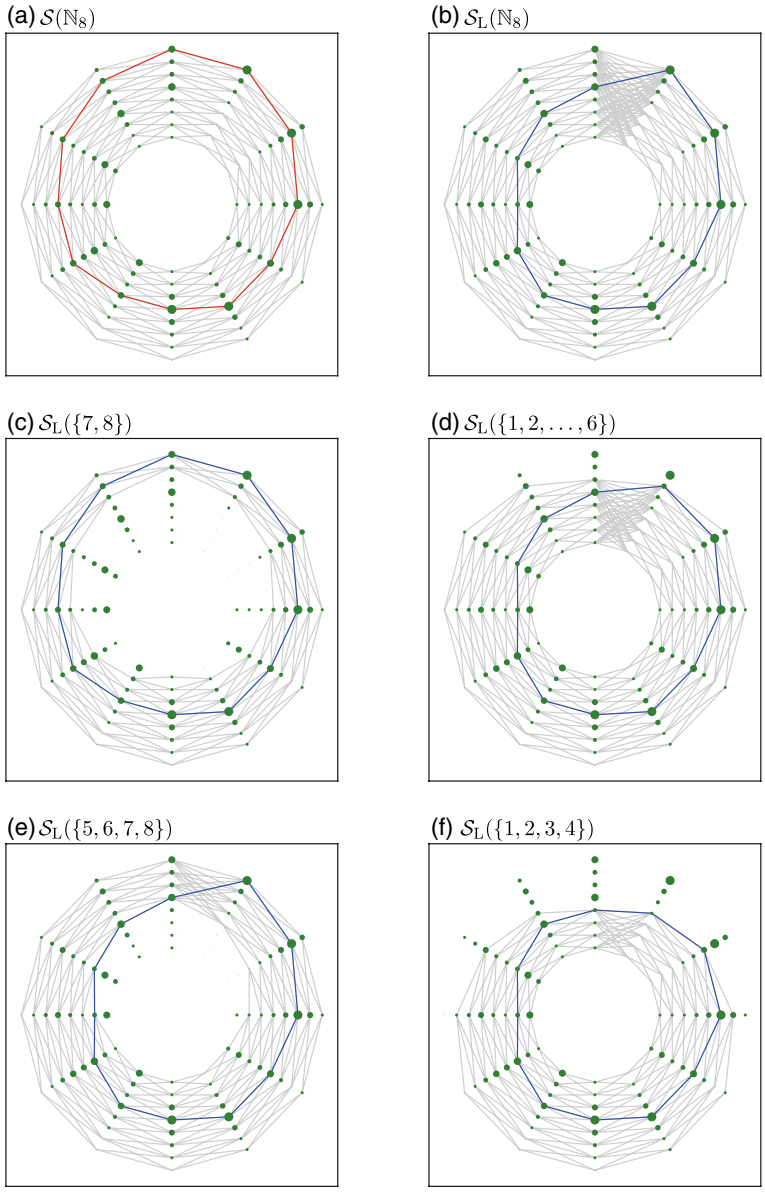

Fig. 5 Relaxed problems in DCDP. Here, a segmentation problem (1) with $(n, m, \varsigma) = (8, 12, 1)$ is considered. We use an m-sided polygon to model the boundary of a glomerulus. The vertices are restricted to be on any of the n points lying on the m rays from the center of the glomerulus. The boundary likeliness is computed on each of the n points, and the configuration that maximizes the sum of the boundary likeliness is found, as described in (1). In Panel (**a**), the sizes of the green circles indicate the quantities of boundary likeliness. As in (1), the feasible configurations of the polygon are restricted to be in $\mathcal{S}(\mathcal{I}_0)$, where $\mathcal{I}_0 = \mathbb{N}_n$. Overlapping all feasible configurations yields the gray edges in Panel (**a**). The optimal polygon is drawn with red edges. DCDP relaxes the feasible region $\mathcal{S}(\mathcal{I}_0)$ to get $\mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)$. In Panel (**b**), the relaxed feasible region $\mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)$ is depicted. The blue polygon is the optimal configuration for the relaxed problem $\boldsymbol{p}_{\mathrm{L},0} = \mathrm{argmax}_{\boldsymbol{p}\in\mathcal{S}_{\mathrm{L}}(\mathcal{I}_0)} J(\boldsymbol{p})$. The proposed algorithm DCDP divides the problem into many sub-problems. The relaxed versions of the four sub-problems with $\mathcal{S}_{\mathrm{L}}(\{7,8\})$, $\mathcal{S}_{\mathrm{L}}(\{1,2,\ldots,6\})$, $\mathcal{S}_{\mathrm{L}}(\{5,6,7,8\})$, and $\mathcal{S}_{\mathrm{L}}(\{1,2,3,4\})$ are illustrated in Panels (**c**), (**d**), (**e**), and (**f**). The blue polygons in (**c**), (**d**), (**e**), and (**f**) are the optimal solutions of the four relaxed problems, respectively. See the main text for details

Results and discussion

In this section, the detection performance is demonstrated by showing the experimental comparisons between S-HOG and R-HOG [10, 11].

As described in the previous section, our method has three stages: pre-screening, segmentation, and classification. Each stage uses its own SVM trained with a hyper-parameter C. In the classification stage, a threshold

θ is used to classify an example; if the SVM score is over the threshold θ, the example is predicted as positive, otherwise, negative. For the pre-screening and classification stages, Set A was used for training SVM, and Set B was used for determining the optimal combination of (C, θ). Sets C, D, and E were used for performance evaluation. SVM for the segmentation stage provides us with the boundary likelihood function. The regularization parameter C for the SVM is determined via the holdout method within Set A. Seventy percent of the glomeruli in Set A are randomly selected for training, and the rest are used for validation. The resulting parameter values were $(C, \theta) = (10, 2)$ for pre-screening, $C = 10$ for segmentation, and $(C, \theta) = (10, -1.5)$ for classification.

Detection performance

Figure 6 illustrates examples of detected glomeruli. In the two images, the candidate glomeruli that passed through pre-screening are depicted with rings that represent the boundaries estimated in the segmentation stage. The numbers printed above the rings are the scores produced by SVM in the classification stage. Candidate glomeruli with SVM scores below $\theta = -1.5$ are excluded from the final detection results. The excluded candidates are depicted with blue rings, and the remaining glomeruli with red rings. It can be observed that non-glomerulus areas are excluded effectively and that true glomeruli are estimated correctly.

For quantitative assessment of detection performance, true positives, false positives, and false negatives have to be defined. True positive glomeruli (TPG) are identified as correctly detected glomeruli, false positive glomeruli (FPG) are wrongly detected glomeruli, and false negative glomeruli (FNG) are the ones that could not be detected. From the definitions of TPG, FPG, and FNG, we can compute for the three widely used performance measures: F-measure, precision, and recall. Precision is the ratio of TPG to the detected glomeruli (i.e. TPG/(TPG + FPG)), recall is the ratio of TPG to the true glomeruli (i.e. TPG/(TPG + FNG)), and F-measure is the harmonic mean of the Precision and Recall.

Figure 7 shows the plots of the F-measure, Precision, and Recall for each testing image. S-HOG achieved an average of 0.866, 0.874, and 0.897 for F-measure, Precision, and Recall, respectively, whereas R-HOG obtained 0.838, 0.777, and 0.911, respectively. While applying detection methods to pathological evaluation, Precision is more important than Recall [11], and in this study, S-HOG achieved considerably higher Precision with a small sacrifice in Recall. A two-sample t-test was performed to assess the statistical differences. While no statistical difference of Recall can be detected (P-value = $3.47 \cdot 10^{-1}$), the differences among F-measure and Precision are significant (P-values = $1.34 \cdot 10^{-3}$ and $3.75 \cdot 10^{-5}$, respectively).

Segmentation performance

Herein, we discuss the performance of the segmentation algorithm. While the main purpose of the proposed method is detection, the proposed DCDP algorithm used for obtaining estimated segmentations may also be applied in some way in studies needing subsequent pathological evaluation [11]. To quantify the accuracy of the estimated areas within the predicted boundaries, 993 annotated glomeruli in Set B were used. True positive area (TPA), false positive area (FPA), and false negative area (FNA) were defined as follows: TPA is the intersection of the true area and estimated area; FPA is the relative complement of the true area in the estimated area; and

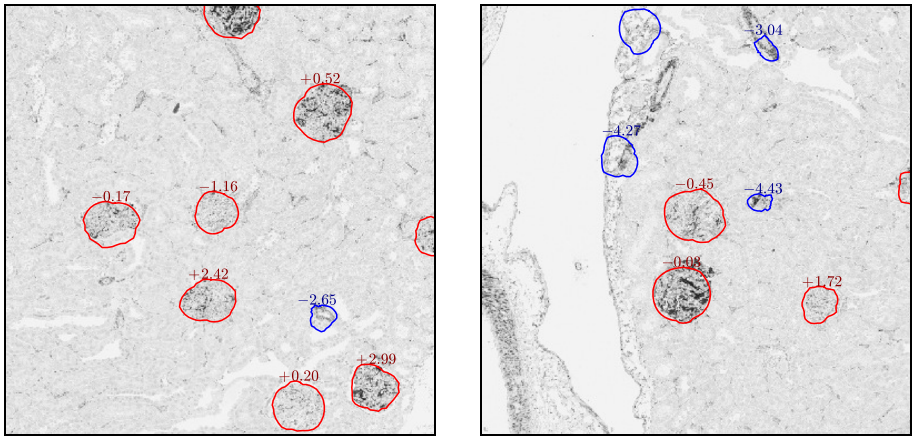

Fig. 6 Examples of detected glomeruli. The estimated boundaries of the glomeruli are depicted with red rings. The areas surrounded with blue rings are passed through the pre-screening stage, but removed in the classification stage. The numbers are the SVM scores resulting from the classification stage. The areas with SVM scores more than $\theta = -1.5$ are classified as a glomerulus. It can be observed that false positives such as vessels detected in the pre-screening stage were successfully removed in the classification stage

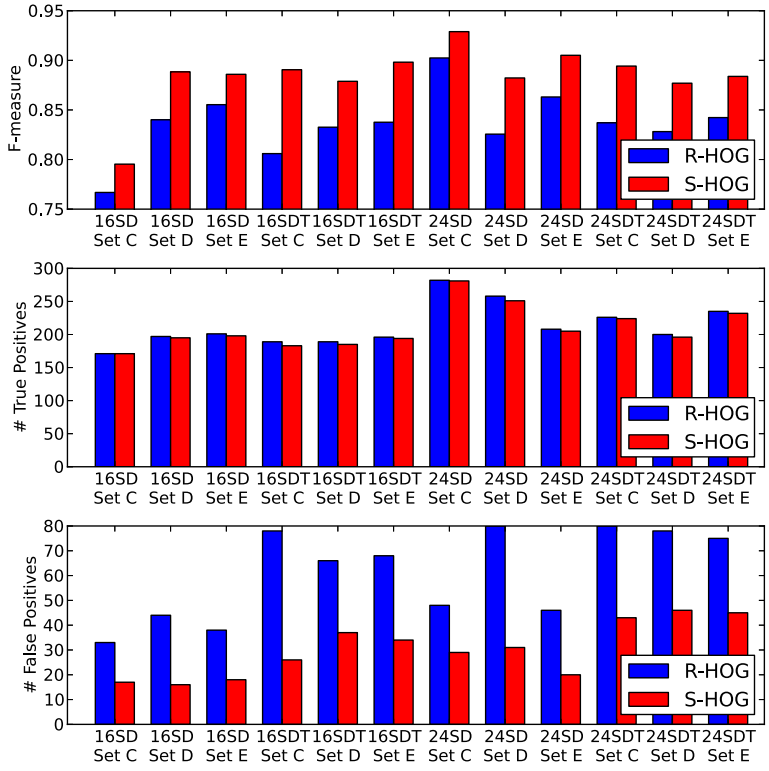

Fig. 7 Detection performances. The proposed descriptor, S-HOG, achieves evident improvement in F-measures compared to the existing descriptor, R-HOG. With small loss of true positives, S-HOG halves false positives of R-HOG. (See subsection on 'Detection performance' for details)

FNA is the relative complement of the estimated area in the true area. For each glomerulus and its estimated area, F-measure, Precision, and Recall can be obtained by counting the pixels in the TPA, FPA, and FNA. The histograms of the F-measure, Precision, and Recall are plotted in Fig. 8, where the frequency is normalized so that the integral is one. Among the glomeruli, 90.1 % are estimated to have F-measures more than 0.8, ensuring reliable assessment of the medicinal effect for drug development.

The computational time of the new segmentation algorithm, DCDP, is compared with that of EDP. The two algorithms solve the same optimization problem, and both algorithms always find the same optimal solution. DCDP and EDP are implemented in C++ language, and the runtimes are measured on a Linux machine with Intel(R) Core(TM) i7 CPU and 8-GB memory. First, the number of times when the $O(nm\varsigma)$ DP routine was invoked, which we denote by n_{dp}, is counted using the annotated glomeruli in Set B. Figure 9a shows the box-plot of n_{dp} for all methods. While the value of n_{dp} for EDP is always n, the values for DCDP depend on the input images and the splitting schemes, Half Split, Max Split, and Adap Split. For 46.32 % of glomeruli, the optimal solutions are found within the first DP routine (i.e. $n_{\mathrm{dp}} = 1$). The medians of the n_{dp}'s when using Half Split, Max Split, and Adap Split are 5, 3, and 3, respectively. In other words, the medians of the depth of the branching tree for each scheme are 3, 2,

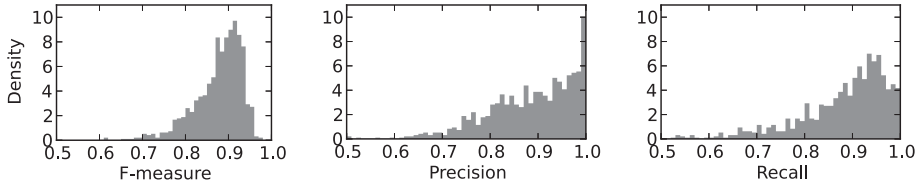

Fig. 8 Segmentation performances. The number of glomeruli are tallied to make a histogram with the F-measure, Precision, and Recall of the pixels on the basis of comparison of true segmentation with estimated segmentation on the *x*-axes. (See 'Segmentation performance' subsection for details)

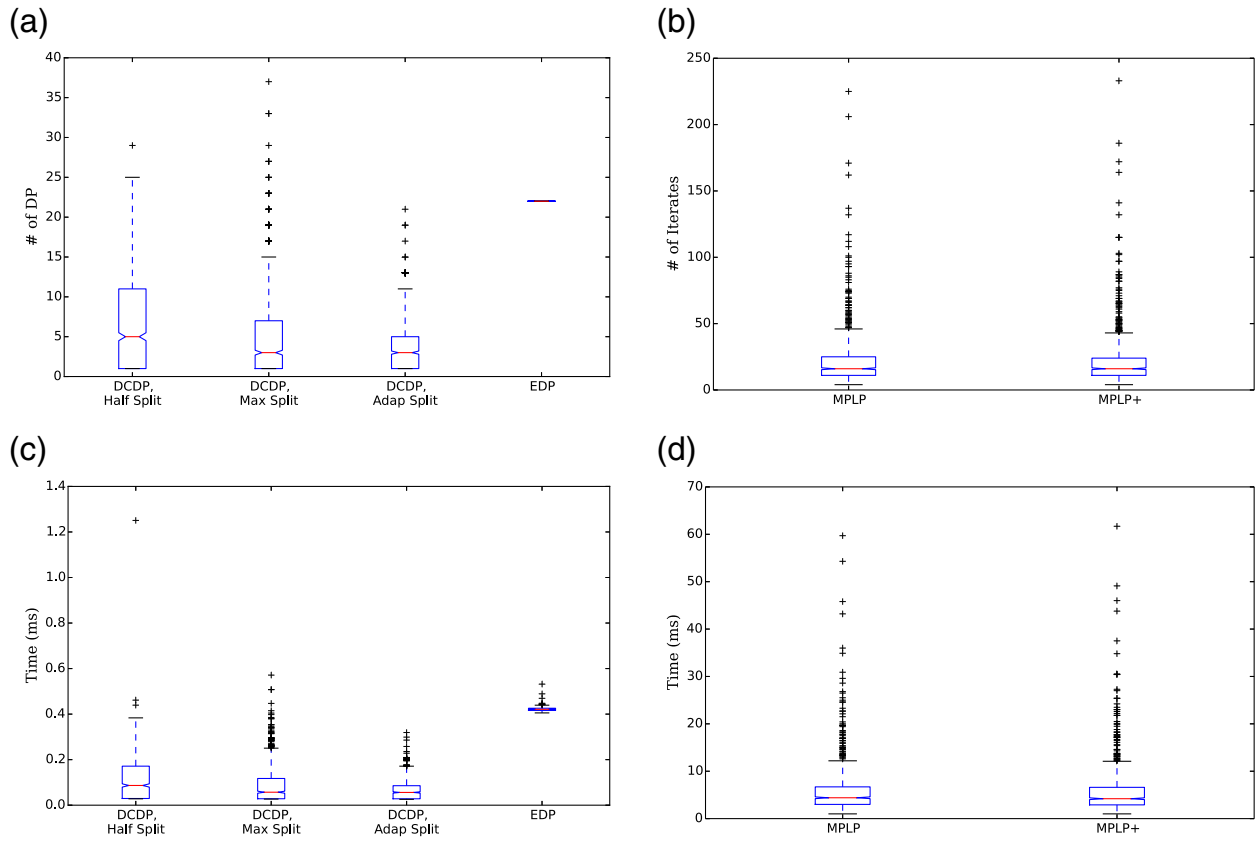

Fig. 9 Runtime comparisons. In Panel (**a**), n_{dp} of DCDP with three splitting schemes and EDP is shown, where n_{dp} is the number of invoking the $O(mn\varsigma)$ DP routine. The number of iterates of MPLP and MPLP+ is plotted in Panel (**b**), where the time complexity for one iterate in MPLP and MPLP+ is $O(mn\varsigma)$, which is equal to one DP routine. The computational time of each algorithm is plotted in Panels (**c**) and (**d**)

and 2, respectively, and the respective 75th percentiles of n_{dp}'s are 11, 7, and 5. For Adap Split, there is no case where n_{dp} is larger than n, whereas the number of glomeruli with $n_{\mathrm{dp}} > n$ are 4 (0.40 %) and 16 (1.61 %) for Half Split and Max Split, respectively. This implies that Adap Split is the best heuristic process among the three splitting schemes.

As considered in 'Methods' section, Adap Split produces the same solution $\boldsymbol{p}_{\mathrm{L}}$ in the branches less frequently when compared to the other schemes. In Half Split and Max Split, the frequency (# of glomeruli) of cases where the solution $\boldsymbol{p}_{\mathrm{L}}$ in the top branch appears again in the second branches is 414 and 314, respectively. These numbers are much larger than the frequency obtained by Adap Split, which is only 97. This explains why Adap Split is faster. The actual runtime of each method is depicted in Fig. 9c, where the medians of the computation times are 0.0866, 0.0570, 0.0560, and 0.418 msec, and the 75th percentiles of the computational times are 0.171, 0.117, 0.0856, and 0.426 msec for Half Split, Max Split, Adap Split, and EDP, respectively. As these values are proportional to the n_{dp}'s, the ratios among the runtimes are almost the same as the ratios among the n_{dp}'s. These results conclude that the proposed algorithm DCDP achieves an exact optimal solution much more efficiently than the existing algorithm EDP while solving the same problem, and that the Adap Scheme is the fastest splitting scheme.

The polygon model employed in this study can be reformulated as an MRF where neighboring vertices have an interaction. This perspective allows us to solve problems (1) by means of algorithms for finding a MAP estimation of MRF models. For comparison with DCDP, we examined the MPLP method [26], which is a state-of-the-art algorithm for MAP estimation of MRF. MPLP is a block coordinate-descent algorithm for minimizing the dual objective of LP relaxation. For our segmentation problem (1), the dual objective is given by

$$\Psi(\boldsymbol{\lambda}) := \sum_{i=1}^{m} \max_{p_i \in \mathbb{N}_m} \left(L_i(p_i) + \lambda(p_i, i, i) + \lambda(p_i, i-1, i)\right) - \sum_{i=1}^{m} \min_{(p_i, p_{i+1}) \in \mathbb{N}_m^2 : |p_i - p_{i+1}| \le \varsigma} \left(\lambda(p_i, i, i) + \lambda(p_{i+1}, i, i+1)\right),$$

where $\boldsymbol{\lambda} := \{\lambda(p_i, i, i) \in \mathbb{R} \mid p_i \in \mathbb{N}_n, i \in \mathbb{N}_m\} \cup \{\lambda(p_{i+1}, i, i+1) \in \mathbb{R} \mid p_i \in \mathbb{N}_n, i \in \mathbb{N}_m\}$ is a set of dual variables, and we used p_{m+1} as the alias of p_1 for simplicity of notation. Both $\lambda(p_1, 0, 1)$ and $\lambda(p_{m+1}, m, m+1)$ are aliases for $\lambda(p_1, m, 1)$. Each variable block in the block coordinate descent is an edge. Hence, our segmentation problem (1) has m variable blocks, and the update rule for i-th edge is given by

$$\lambda(p_i, i, i) := -\frac{1}{2}\left(L_i(p_i) + \lambda(p_i, i-1, i)\right) + \frac{1}{2}\max_{p_{i+1} \in [p_i - \varsigma, p_i + \varsigma]} \left(L_{i+1}(p_{i+1}) + \lambda(p_{i+1}, i, i+1)\right),$$

and

$$\lambda(p_{i+1}, i, i+1) := -\frac{1}{2}\left(L_{i+1}(p_{i+1}) + \lambda(p_{i+1}, i+1, i+1)\right) + \frac{1}{2}\max_{p_i \in [p_{i+1} - \varsigma, p_{i+1} + \varsigma]} \left(L_i(p_i) + \lambda(p_i, i, i)\right).$$

The time complexity of one iterate is $O(nm\varsigma)$, which is equal to that of Algorithm 2, a dynamic program for solving each sub-problem used in DCDP. Typically, larger variable blocks reach the convergence faster in the block coordinate-descent algorithm. It can be seen easily from the update rule of i-th edge that the dual variables of $(m/2)$ odd-numbered edges are updated simultaneously and those of $(m/2)$ even-numbered edges are updated simultaneously. Then, the number of blocks is reduced to two. This algorithm is referred to as *MPLP+*. We actually implemented both MPLP and MPLP+ in C++ language and applied it to each of the 993 glomeruli images. MPLP and MPLP+ successfully obtained the optimal solutions for all the images, although MPLP and MPLP+ are not guaranteed theoretically to achieve the optimal solution. The number of iterates and the computational times are depicted in Fig. 9b and d, respectively. Although MPLP+ has larger variable blocks than MPLP, it did not significantly improve convergence. Furthermore, it turned out that both methods are too slow to be compared with DCDP.

Conclusions

In this study, a new descriptor, Segmental HOG, was proposed for specific glomeruli detection in microscopy images. The descriptor was based on the boundary of the glomeruli to acquire robustness for variations in intensities, sizes, and shapes. A new segmentation algorithm, DCDP, was developed to locate the boundary of possible glomeruli. Empirical results show significant improvement compared to the state-of-the-art descriptor, Rectangular HOG, for the task of glomerulus detection in microscopy images. Moreover, experimental results reveal that DCDP is much faster than the existing segmentation algorithm EDP.

Several possible uses of the proposed method can be considered. For instance, an appropriate size of the sliding window should be chosen if the proposed method is applied to microscopic images with different resolutions. In addition, while the boundary likeliness function is the same for any direction in the segmentation algorithm, different boundary likeliness functions can be used for detecting other organs depending on the orientation. For the block division of the S-HOG descriptor, 24 blocks were used in this study, as depicted in Fig. 4c, but a different number of blocks with a different division can be used for other applications. Future studies include exploring such extensions in other applications of Segmental HOG.

Endnote

[1] Sliding windows are used in both pre-screening and segmentation.

Abbreviations

R-HOG: Rectangular histogram of oriented gradients; S-HOG: Segmental histogram of oriented gradients; EDP: Exhaustive dynamic program; DCDP: Divide and conquer dynamic program; DP: Dynamic program; SVM: Support vector machine; SDT: Spontaneously diabetic torii; SD: Sprague-Dawley; 16SD: 16-week-age SD rat; 16SDT: 16-week-age SDT rat; 24SD: 24-week-age SD rat; 24SDT: 24-week-age SDT rat; TPG: True positive glomeruli; FPG: False positive glomeruli; FNG: False negative glomeruli; TNG: True negative glomeruli; TPA: True positive area; FPA: False positive area; FNA: False negative area.

Competing interests

Tetsuhiro Kakimoto and Kinya Okada work for Mitsubishi Tanabe Pharma Corporation.

Authors' contributions

TKat and RR proposed the algorithm and designed the experiments. TKat, HN, and YH wrote the computer codes. HN and TKat performed the experiments. OT joined discussion. TKak, and KO prepared the materials. TKat, RR, and KO drafted the manuscript. All authors read and approved the final manuscript.

Acknowledgements

This research received no specific funding from any agency.

Author details

[1] Faculty of Science and Engineering, Gunma University, Kiryu-shi, Gunma 376–8515, Japan. [2] Center for Informational Biology, Ochanomizu University, Bunkyo-ku, Tokyo 112-8610, Japan. [3] Gunma University Hospital, Maebashi-shi, Gunma 371–8511, Japan. [4] Research Division, Mitsubishi Tanabe Pharma Corporation, Toda-shi, Saitama 335–8505, Japan.

Received: 9 January 2015 Accepted: 11 September 2015
Published online: 30 September 2015

References

1. Zhang J, Hu J. Glomerulus extraction by optimizing the fitting curve. In: ISCID, vol. 1. IEEE; 2008. p. 169–72.
2. Ma J, Zhang J, Hu J. Glomerulus extraction by using genetic algorithm for edge patching. In: IEEE CEC. Trondheim, Norway: IEEE; 2009. p. 2474–9.
3. Hughson MD, Puelles VG, Hoy WE, Douglas-Denton RN, Mott SA, Bertram JF. Hypertension, glomerular hypertrophy and nephrosclerosis: the effect of race. Nephrol Dial Transplant. 2013;29(7):1399–409.
4. Rasch R, Lauszus F, Thomsen JS, Flyvbjerg A. Glomerular structural changes in pregnant, diabetic, and pregnant-diabetic rats. APMIS. 2005;113(7-8):465–72.
5. Munder S, Gavrila DM. An experimental study on pedestrian classification. IEEE Trans Pattern Anal Mach Intell. 2006;28(11):1863–8.

6. Maji S, Berg A, Malik J. Classification using intersection kernel support vector machines is efficient. In: IEEE CVPR. Anchorage, AK: IEEE; 2008. p. 1–8.
7. Papageorgiou C, Poggio T. A trainable system for object detection. Int J Comput Vision. 2000;38(1):15–33.
8. Viola P, Jones MJ. Robust real-time face detection. Int J Comput Vision. 2004;57(2):137–54.
9. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: IEEE CVPR, vol. 1. Washington, DC, USA: IEEE Computer Society; 2005. p. 886–93.
10. Hirohashi Y, Relator R, Kakimoto T, Saito R, Horai Y, Fukunari A, et al. Automated quantitative image analysis of glomerular desmin immunostaining as a sensitive injury marker in spontaneously diabtic torii rats. J Biomed Image Process. 2014;1(1):20–8.
11. Kakimoto T, Okada K, Hirohashi Y, Relator R, Kawai M, Iguchi T, et al. Automated image analysis of a glomerular injury marker desmin in SDT rats treated with losartan. J Endocrinol. 2014;222(1):43–51.
12. Kakimoto T, Okada K, Fujitaka K, Nishio M, Kato T, Fukunari A, et al. Quantitative analysis of markers of podocyte injury in the rat puromycin aminonucleoside nephropathy model. Exp Toxicol Pathol. 2015;67(2): 171–7.
13. Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vision. 2004;60(2):91–110.
14. Ahonen T, Hadid A, Pietikainen M. Face description with local binary patterns: application to face recognition. IEEE Trans Pattern Anal Mach Intell. 2006;28(12):2037–41.
15. Jiang H, He B, Fang D, Ma Z, Yang B, Zhang L. A region growing vessel segmentation algorithm based on spectrum information. Comput Math Methods Med. 2013. http://dx.doi.org/10.1155/2013/743870.
16. Cremers D, Rousson M, Deriche R. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. Int J Comput Vision. 2007;72(2):195–215.
17. Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models. Int J Comput Vision. 1988;1(4):321–31.
18. Wang T, Cheng I, Basu A. Fluid vector flow and applications in brain tumor segmentation. IEEE Trans Biomed Eng. 2009;56(3):781–9.
19. Xie X. Active contouring based on gradient vector interaction and constrained level set diffusion. IEEE Trans Image Process. 2010;19(1): 154–64.
20. Ciresan D, Giusti A, Gambardella LM, Schmidhuber J. Deep neural networks segment neuronal membranes in electron microscopy images In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. NIPS. Curran Associates, Inc; 2012. p. 2852–60.
21. Kvarnström M, Logg K, Diez A, Bodvard K, Käll M. Image analysis algorithms for cell contour recognition in budding yeast. Opt Express. 2008;16(17):12943–57.
22. Rozanov JA, Elson CM. Markov random fields. New York: Springer; 1982.
23. Viterbi A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans Inf Theor. 2006;13(2):260–269.
24. Yanover C, Meltzer T, Weiss Y. Linear programming relaxations and belief propagation - an empirical study. J Mach Learn Res. 2006;7:1887–907.
25. Weiss Y, Freeman WT. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. IEEE Trans Inf Theor. 2006;47(2):736–44.
26. Globerson A, Jaakkola TS. Fixing max-product: Convergent message passing algorithms for MAP lp-relaxations. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007. Cambridge MA: MIT Press; 2007. p. 553–60.
27. Schölkopf B, Smola AJ. Learning with Kernels. Cambridge, MA: MIT Press; 2002.
28. Shawe-Taylor J, Cristianini N. Kernel Methods for Pattern Analysis. Cambridge, MA: Cambridge University Press; 2004.