Fast Ray Features for Learning Irregular Shapes

Kevin Smith[1] Alan Carleton[2] Vincent Lepetit[1]
[1]Computer Vision Laboratory
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
[2]Department of Basic Neurosciences
Medical Faculty, University of Geneva
CH-1211 Genève, Switzerland

Abstract

We introduce a new class of image features, the Ray feature set, that consider image characteristics at distant contour points, capturing information which is difficult to represent with standard feature sets. This property allows Ray features to efficiently and robustly recognize deformable or irregular shapes, such as cells in microscopic imagery. Experiments show Ray features clearly outperform other powerful features including Haar-like features and Histograms of Oriented Gradients when applied to detecting irregularly shaped neuron nuclei and mitochondria. Ray features can also provide important complementary information to Haar features for other tasks such as face detection, reducing the number of weak learners and computational cost.

Ray features can be efficiently precomputed to reduce cost, just as precomputing integral images reduces the overall cost of Haar features. While Rays are slightly more expensive to precompute, their computational cost is less than that of Haar features for scanning an AdaBoost-based detector window across an image at run-time.

1. Introduction

Cascaded Adaboost classifiers, introduced in [14], are a principled and efficient method of detecting objects in images. The success of this approach has inspired a large body of derivative work, though much of it focuses on the classifier itself [13]. By comparison, a much smaller body of work focuses on alternatives to the powerful Haar-like features proposed in [14], which are attractive for their use of integral images to minimize evaluation cost. This is unfortunate because Haar features, as well as Histograms of Oriented Gradients (HOG), appear to be inefficient at detecting highly deformable objects such as biological cells.

Our motivation for this work is the task of detecting mitochondria and neuron nuclei in microscopic imagery, which can deform drastically as depicted in Fig. 1. Haar and HOG based weak learners are not particularly well suited to

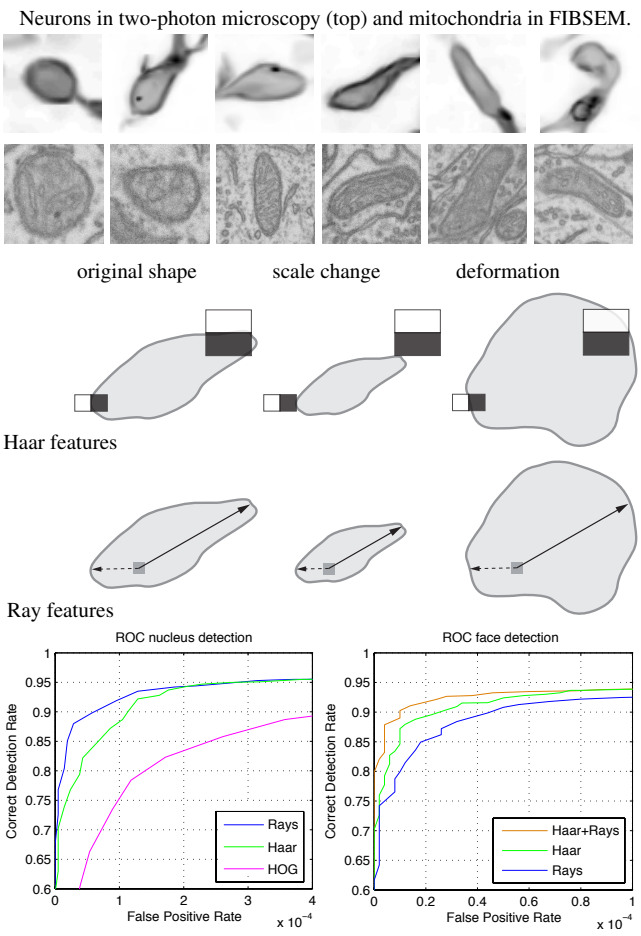

Figure 1. Neuron nuclei *(row 1)* and mitochondria *(row 2)* can vary dramatically in shape. Irregularities such as a scale change or deformation can cause a large change in the response of conventional Haar-like features *(row 3)*. To characterize the range of possible shapes, additional Haar-like features are required. Ray features *(row 4)* describe the shape more compactly and robustly by comparing the relative distance from a central location to the nearest contours in two directions. Experiments show Rays outperform an equal number of HOG and Haar features for nucleus detection *(bottom left)*. Rays can also be combined with Haar features to improve face detection performance *(bottom right)*.

this task because they depend on reliable cues defined at precise locations, such as the nose appearing in the center of a face. While there is some tolerance for small variations, large deviations in the training set require additional weak learners to represent the range of variations. If the training set does not encompass the full range of deformations, the classifier will exhibit poor performance.

To overcome these limitations we introduce the *Ray feature set*, which is comprised of four variants of a basic function shown in Fig. 2. Rays exploit the observation that while we cannot predict the precise locations of characteristic image features for highly deformable objects, we can predict their locations relative to one another, or relative to certain other locations. For example, we can reliably predict the relative distance to the edges of a cell from the center of the cell, as seen in Fig. 1. Rays make use of this principle by testing an image property at the nearest contour in some direction, given an image location. This allows the classifier to test for the presence of occluding contours, which are important cues for object recognition but difficult to capture with other features.

The benefits of Ray features are not limited to deformable blob-like objects. While they cannot replace Haar or HOGs on classic problems such as face detection, Rays can improve detection performance when used in conjunction with these features. Furthermore, Ray features are efficient to compute. Like integral images, Ray features can be precomputed for the entire image allowing them to be evaluated at run-time at a constant cost.

In the remainder of the paper, we first discuss related work, we then describe the Ray feature set, and finally compare it against HOG and Haar-like features for microscopy detection and face detection tasks.

2. Related Work

The success of Haar wavelets, efficiently evaluated using integral images [14], has invited a host of incremental improvements. These improvements allow for limited [8] and arbitrary [5] rotations, or employ co-occurrences of multiple Haar-like features [10]. Others have sought to augment Haar-like learners with edge orientation histogram learners [5] to detect faces and persons, but these learners did not prove powerful enough to be used without Haar learners. Rather than applying Haar features directly to the image, others have applied them to gradient magnitudes to detect persons [12]. However, HOGs have been shown to perform well at detecting persons in images [3].

Ray features share commonalities to techniques used by the shape recognition community [1, 2, 6] even if our context is more general: These techniques are usually applied to binary images already centered on the object to be recognized, while we parse grayscale images to find the objects of interest. In [6], local descriptors are defined by the distances from a given contour point to the neighboring contour points. Such a descriptor can be used for shape recognition with a nearest-neighbor classifier, for example. A Ray learner is more general because it considers distances from an arbitrary location to neighboring contours. The tag arrangements of [1] are very close to one of our features ($f_i^{[\text{diff}]}$, see Section 3.1.1). Our feature is more flexible because the distance between the contour parts can be learned, and again, we can consider non-centered grayscale images. Moreover there is no obvious efficient adaptation of these methods for use in a scanning detector. Because Ray features can be computed very quickly, many features can be used to train a boosted classifier for detection.

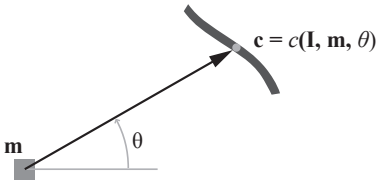

Figure 2. Function $c(\mathbf{I}, \mathbf{m}, \theta)$. This function is used to define the Rays feature set. For a given location $\mathbf{m}$, it returns the location of the closest edge point $\mathbf{c}$ in the direction stated by angle θ.

Another related technique is the use of the Distance Transform, where objects are recognized by computing the sum of distances to contours along templates [4]. However, these approaches usually rely on nearest neighbor classification, while an Adaboost classifier trained with Ray features provides better speed and reliability. Moreover, the four varieties of Ray features provide a richer image description.

Rays can be applied to many detection problems in which contours are useful and shapes can deform. However, our target application is the detection of cells in microscopy data. Others have applied machine learning techniques to this task for less complex types of (typically cultured) cells, meaning the background is relatively clean and cell shapes are simple in comparison with our data. Examples include neural networks applied to detect white blood cells [15] or fluorescent marked lymphocytes [11], and Support Vector Machines used to detect lymphoma cells [9] and tuberculosis bacteria [7].

3. The Ray Feature Set

In this section, we describe the Ray feature set and provide an intuitive explanation for their discriminative power. We then show how they can be efficiently precomputed, and compare their computational complexity to Haar features.

3.1. Description

The Ray feature set contains four distinct image features. Each exploits the observation while we cannot easily predict the locations of distinguishing image features for deformable objects, we can predict their locations relative to

other certain locations, or to one another. Given an image location $\mathbf{m}$ and direction θ, Ray features test some property of the nearest edge location in direction θ, as depicted in Fig. 2. This distinguishes Rays from other features since the part of the image that is actually tested—the closest edge point—can be distant from the starting image location. More importantly, for certain Ray features this distance may change without changing the feature response. This property gives Ray features tolerance to shape variations learned during training.

The Ray feature set, depicted Fig. 3, contains four features, denoted by $f_i^{[\text{diff}]}$, $f_i^{[\text{dist}]}$, $f_i^{[\text{ori}]}$, and $f_i^{[\text{norm}]}$. Each of these features depend on the function:

$$\mathbf{c} = c(\mathbf{I}, \mathbf{m}, \theta) \, ,$$

which returns the location $\mathbf{c}$ of the closest contour point in image $\mathbf{I}$ to location $\mathbf{m}$ in the direction defined by angle θ.

3.1.1 Distance Difference Feature: $f_i^{[\text{diff}]}$

The first and most dominant feature, $f_i^{[\text{diff}]}$, compares the relative distances from a given location to the nearest contours in two search directions. It is defined by:

$$f_i^{[\text{diff}]}(\mathbf{I}) = \frac{\|c(\mathbf{I}, \mathbf{m}_i, \theta_i) - \mathbf{m}_i\| - \|c(\mathbf{I}, \mathbf{m}_i, \theta_i') - \mathbf{m}_i\|}{\|c(\mathbf{I}, \mathbf{m}_i, \theta_i) - \mathbf{m}_i\|} \, ,$$

with parameters image location $\mathbf{m}_i$ and two angles, θ_i and θ_i'. Shown in Fig. 3, $f_i^{[\text{diff}]}(\mathbf{I})$ computes the difference of distances from $\mathbf{m}_i$ to the edge points found in directions given by θ_i and θ_i'. This difference value is normalized by the first distance a, providing invariance to scale changes [1].

The advantage of $f_i^{[\text{diff}]}(\mathbf{I})$ over Haar features is depicted in the third and fourth rows of Fig. 1. When the object is scaled or deformed, the response of Haar features which characterize the original object change rapidly. This means additional Haar learners would be required to represent such variations. The response of $f_i^{[\text{diff}]}(\mathbf{I})$, on the other hand, does not change in these examples, meaning a single Ray feature can reliably represent a family of similar shapes.

3.1.2 Distance Feature: $f_i^{[\text{dist}]}$

The distance feature considers the absolute distance to the closest edge point in a given direction:

$$f_i^{[\text{dist}]}(\mathbf{I}) = \|c(\mathbf{I}, \mathbf{m}_i, \theta_i) - \mathbf{m}_i\| \, .$$

This feature is parametrized by an image location $\mathbf{m}_i$ and a direction defined by θ_i. It complements the previous scale-invariant feature, $f_i^{[\text{diff}]}$, by providing some some absolute constraints on object scale, as complete invariance to scale change is not always desirable.

[1] When classifying, it is more efficient to multiply the decision threshold τ_i by a, skipping the division operation. This eliminates the possibility of dividing by zero, and uses a less costly multiplication operation.

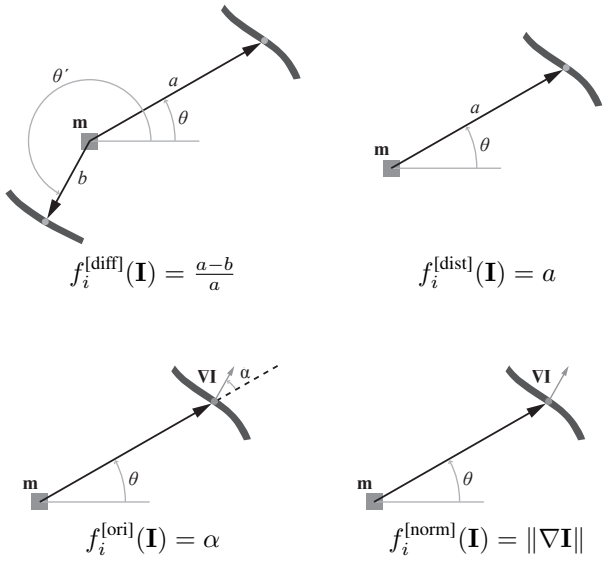

Figure 3. The set of four Ray features. See text for details.

3.1.3 Orientation Feature: $f_i^{[\text{ori}]}$

The orientation feature considers the orientation of the nearest contour in direction θ. It computes the dot product of the search direction and the gradient at the contour location:

$$f_i^{[\text{ori}]}(\mathbf{I}) = \frac{\nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}_i, \theta_i))}{\|\nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}_i, \theta_i))\|} \cdot (\cos\theta_i, \sin\theta_i)^\top \, ,$$

where $\nabla \mathbf{I}(\mathbf{c})$ denotes the gradient of image $\mathbf{I}$ at $\mathbf{c}$. This feature characterizes objects by their contour orientations. It can be useful for recognizing convex closed shapes from concave or open shapes. When $\mathbf{m}_i$ is in the center of a closed shape, $f_i^{[\text{ori}]}$ often returns values close to 1. Combining a few of these features can test its closedness.

This feature is related to HOG [3] in the sense that both methods characterize gradient orientation based on location. HOGs accomplish this through local histograms, while Rays search for contours with distinctive orientations using reliable points located some distance away.

3.1.4 Norm Feature: $f_i^{[\text{norm}]}$

The norm feature considers the gradient strength of the nearest contour in direction θ:

$$f_i^{[\text{norm}]}(\mathbf{I}) = \|\nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}_i, \theta_i))\| \, .$$

As this is the only Ray feature that tests the image intensity, it is useful for characterizing the appearance of the contours of an object.

3.1.5 Ray Weak Learners

Each Ray feature can easily be used to define a weak learner in the form of a decision stump h_i to train a boosted classi-

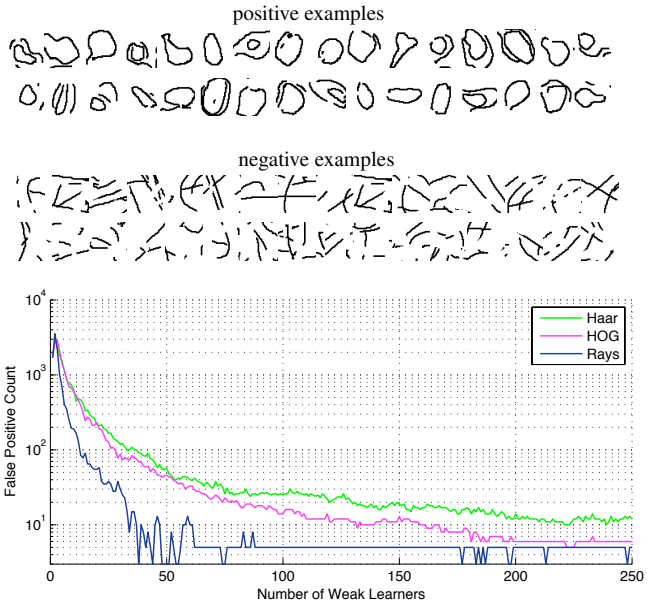

Figure 4. Recognizing irregular cell-like contours. Ray features are compared against Haar and HOG on a data set of hand-drawn shapes. The training and test sets each contain 1,200 cell-like contours and 20,000 non cell-like shapes. For each feature set, we trained a single stage AdaBoost classifier. The detection rate was set at 90%, allowing the False Positive rate to vary. Results on the test set, shown above, show that Rays outperform the other features, achieving a better overall recognition rate, and requiring less weak learners for a given performance level.

fier using a method such as AdaBoost:

$$h_i(\mathbf{I}) = \begin{cases} 1 & \text{if } p_i f_i^{[\cdot]}(\mathbf{I}) < p_i \tau_i \\ -1 & \text{otherwise} \end{cases} ,$$

where p_i is polarity, $f_i^{[\cdot]}(\mathbf{I})$ the response of a Ray feature, and τ_i is a decision threshold.

3.2. Discriminative Power of the Ray Features

To demonstrate their discriminative power, we applied Ray features to two data sets of hand-drawn objects that exhibit characteristics representative of living cells.

In the first data set, the task was to recognize cell-like shapes from non-contours. Representative samples are shown Fig. 4. The positive class contains cell-like contours, and the negative class corresponds to noisy portions of curves. We compare the results we obtained using an AdaBoost classifier with either Ray, Haar, or HOG features. The results are presented in the plot of Fig. 4, where the log-scale y-axis corresponds to the number of false alarms made by the classifier, while the x-axis corresponds to the number of weak learners in the classifier. For clarity of presentation, we used a single-stage classifier. Later experiments use a cascaded classifier. The results show that for a given number of learners, Rays outperform the other features.

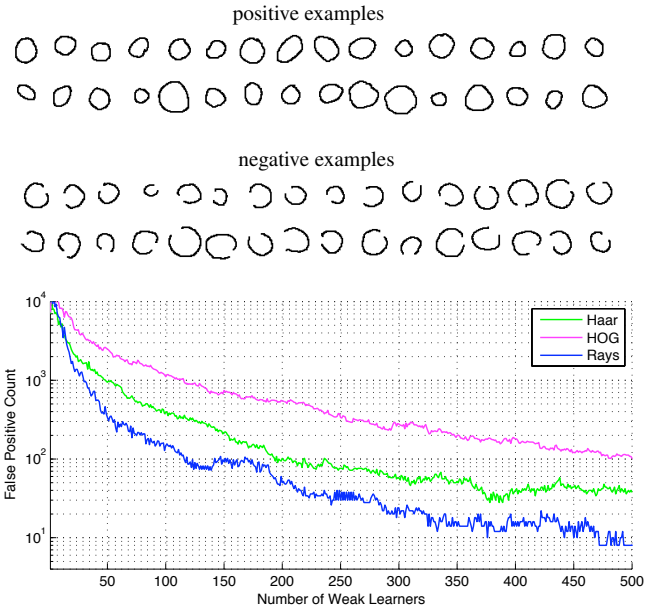

Figure 5. Recognizing closed from open contours. The positive class contains 1,200 closed contours, and the negative class contains 10,000 open contours. Other test conditions are similar those of the experiment shown Fig. 4. Ray features outperform the Haar and HOG features on this task as well.

The second task was to recognize closed contours from open contours. Similar experimental conditions were used, with 1,200 positive and 10,000 negative samples in the training and test sets, seen in Fig. 5. Results appear in the plot of Fig. 5, showing that Ray features are better able to recognize closed contours from open contours.

3.3. Efficient Precomputation

Ray features can be efficiently precomputed for an entire image, making them extremely inexpensive when searching for objects by scanning a detector window. This is analogous to the practice of precomputing integral images in order to dramatically speed up the run-time computation of Haar features. While the preprocessing cost of Ray features is slightly larger than that of Haar features, this cost is recovered at run-time as precomputed Ray features are cheaper than Haar features computed from an integral image. This is described in more detail in Section 3.4.

Rays are precomputed by extracting a binary edge map $\mathbf{B}$ from the input image, and then scanning $\mathbf{B}$ at regular angular intervals $\Theta = \{\theta_1, ..., \theta_i, ..., \theta_q\}$ while computing simultaneously $f_i^{[\text{dist}]}$, $f_i^{[\text{ori}]}$, and $f_i^{[\text{norm}]}$. The difference feature, $f_i^{[\text{diff}]}$, cannot easily be precomputed. Instead, it is calculated from $f_i^{[\text{dist}]}$ at run-time, though the cost is still small. Pseudo-code to precompute Ray features is given in Fig. 6.

In our experiments, the angular interval is typically 30° as it is simple to write efficient code to scan the image at this interval. As depicted in Fig. 7, features at a location along

Algorithm 1 Precomputation of Ray features

Set initial values:
  set scanline direction $\mathbf{u} = (\cos\theta, \sin\theta)^\top$
  initial distance $d \leftarrow 0$
  initial orientation $o \leftarrow 1$
  initial contour norm $n \leftarrow 0$
`For` each location $\mathbf{m}_i \in S$
  set $f_i^{[\text{dist}]}(\mathbf{m}_i) = d$
  set $f_i^{[\text{ori}]}(\mathbf{m}_i) = o$
  set $f_i^{[\text{norm}]}(\mathbf{m}_i) = n$
  `if` $\mathbf{m}_i$ lies on an edge in $\mathbf{B}$:
    $d \leftarrow 0$
    $n \leftarrow \|\nabla\mathbf{I}(\mathbf{m}_i)\|$
    $o \leftarrow \frac{1}{n}\nabla\mathbf{I}(\mathbf{m}_i) \cdot \mathbf{u}$
  `else`
    $d \leftarrow d + 1$

Figure 6. Pseudo-code to efficiently pre-compute Ray features $f_i^{[\text{dist}]}$, $f_i^{[\text{ori}]}$, and $f_i^{[\text{norm}]}$ given a scan line S at orientation θ. For efficiency, we approximate the norm $\|\mathbf{c} - \mathbf{m}_i\|$ in $f_i^{[\text{dist}]}$ by incrementing d for each step in S. Expensive gradient norm and orientation features are only computed when an edge is crossed.

the scan line $\mathbf{m}_i$ can be computed by recalling the gradient at the last contour the scan line crossed, and by counting the steps taken since crossing the last contour.

Because Ray features depend on meaningful edges, the choice of method to extract $\mathbf{B}$ is important. We have found Sobel filters to be a good compromise between efficiency and edge quality. In practice, it is useful to define additional sets of Rays features corresponding to different edge threshold settings, or other edge extraction methods.

3.4. Computational Cost

While the precomputation cost of Ray features is slightly greater than that of Haar features due to edge extraction and multiple image scans corresponding to Θ, Rays are less expensive at run-time when classifying a scanning detector window. The most simple Haar feature computes the intensity difference of two regions in a scan window. Using integral images, this requires six memory accesses, five addition/subtraction operations, and two multiplications by 2. In contrast, Ray features $f_i^{[\text{dist}]}$, $f_i^{[\text{ori}]}$, and $f_i^{[\text{norm}]}$ only require a single memory access, while $f_i^{[\text{diff}]}$ requires two memory accesses, one subtraction, and one multiplication[2]. Implemented on a 2.4GHz MacBook Pro, scanning times are summarized below for a 24×24 detector window on a 640×480 image with $\sigma = 1.5$ (273,150 total windows):

feature	precomputation	run-time	total
Rays	85 ms	2,280 ms	2.4 s
Haar	1.2 ms	7,304 ms	7.3 s

[2]Modern CPUs process addition and multiplication at the same speed.

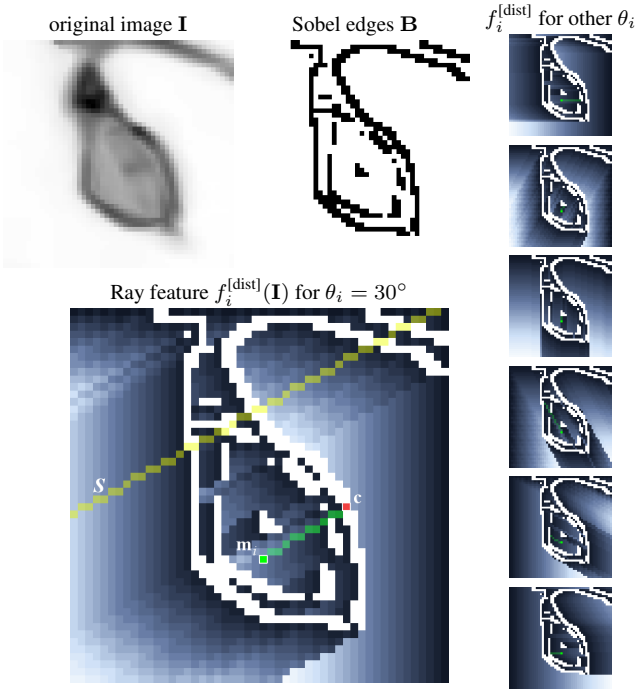

Figure 7. Efficient computation of Ray features. A binary edge mask $\mathbf{B}$ and gradient approximation $\nabla\mathbf{I}$ are extracted using standard techniques such as a Sobel filter. As the image is scanned along the scan line S (yellow), $f_i^{[\text{dist}]}(\mathbf{m}_i)$ is approximated as the number of steps along the scanline from $\mathbf{c}$ to $\mathbf{m}_i$. Note that $\mathbf{m}_i$ does not lie on S for clarity of presentation. Dark intensities correspond to short distances, bright ones to long distances. Orientation $f_i^{[\text{ori}]}(\mathbf{m}_i)$ and norm $f_i^{[\text{norm}]}(\mathbf{m}_i)$ features correspond to values computed at $\mathbf{c}$, and are only updated when a contour is crossed.

4. Results

In this section, we first provide implementation details related to the AdaBoost classifier and each feature type. We then discuss the data sets used in our experiments and provide results for Ray, Haar, and HOG features on each data set. Finally, we discuss the implications of the results.

4.1. Implementation Details

Our experiments compare the classification power of Ray features to Haar and HOG features. Scanning detectors were built using cascaded AdaBoost classifiers for each feature type, trained as described in Table 2 of [14]. Each stage in the detector contains an AdaBoost classifier which rejects some negative samples while passing on positive samples to the next, more discriminative stage. Each detector shared a common schedule of goals defining the false positive rate F_i and detection rate D_i for each stage, yielding the ultimate goals: $F = 10^{-5}$ and $D = .9$. The classifiers were trained using same image sets scaled to a standard 24×24 detector window size. Detection results were achieved by scanning the detector over the test images in 2 pixel steps at

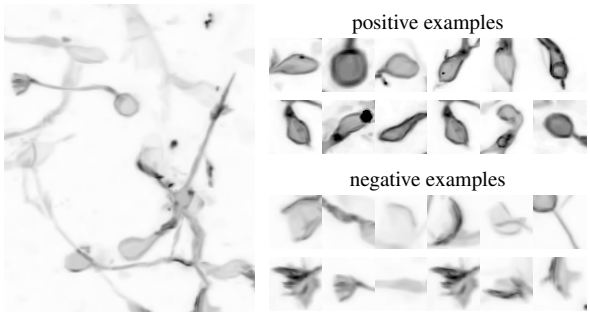

Figure 8. Nuclei data set. *(left)* Maximum intensity projection of 2-photon microscopy image containing fluorescent marked immature neurons migrating in an adult mouse brain at the μm scale. *(right)* training examples. Detecting nuclei is difficult due to distracting objects including growth cones, neurites, and dead cells.

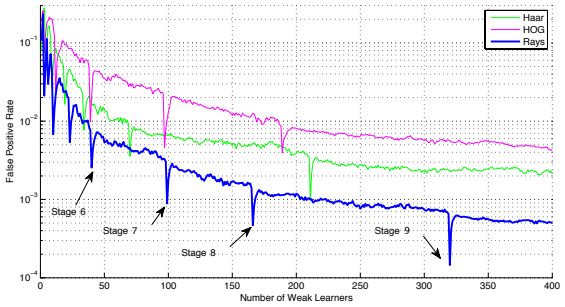

Figure 9. Training AdaBoost to recognize nuclei. This plot shows classification results obtained on the validation set during training. Ray features allow an AdaBoost classifier to obtain better detection and false alarm rates for a given number of weak learners. Periodic downward spikes correspond to a new cascade stage, where some learners are spent reaching the stage's detection rate goal D_i, and causing the false alarm rate to temporarily drop.

scales separated by a $\sigma = 1.5$ scale factor.

Ray features were defined over a regular angular interval of $\theta_i = 30°$. $f_i^{[\text{diff}]}$ angle pairs were defined as the set of unique combinations of these angles. Rays were defined at every second pixel location in the scan window. Various thresholds were used to obtain $\mathbf{B}$; five threshold levels chosen for each data set. For a 24×24 window, 47,520 $f_i^{[\text{diff}]}$, 8,640 $f_i^{[\text{dist}]}$, 8,640 $f_i^{[\text{ori}]}$, and 8,640 $f_i^{[\text{norm}]}$ weak learners were defined, for a total of 73,440 Ray learners.

Haar-like features were implemented as described in [14], using integral images to compute intensity differences between vertical, horizontal, three-rectangle, and four-rectangle features. A total of 95,251 haar-like weak learners were defined for the 24×24 window.

Histogram of oriented gradients (HOG) features were implemented as in [3], except 8×8 pixel blocks of 4×4 pixels were used, due to the small 24×24 window size. A simple $[-1, 0, 1]$ filter is used to extract the gradient. Linear gradient voting was used on 9 orientation bins from $0°$ to $180°$. Weak learners were built from HOG histogram elements, for a total of 1,296 HOG weak learners.

4.2. Experiments

We tested the capability of Ray features related to three tasks: detecting the nuclei of migrating neurons in 2-photon microscopic imagery, detecting mitochondria in focused ion beam scanning electron microscopy (FIBSEM), and on the more common task of detecting faces in photographs. For each task, we provide two means of comparison. First, we fixed the number of weak learners and compared the detection D_i and false alarm F_i performance of each feature type. For this comparison, we provide ROC curves in Fig. 1. Another method of comparison is to investigate how many weak learners are required to achieve a similar performance level, which is conveniently provided by the common stage goals. Results in Fig. 13 provide the number of learners required for a given stage, indicating the amount of learners needed for a similar performance level.

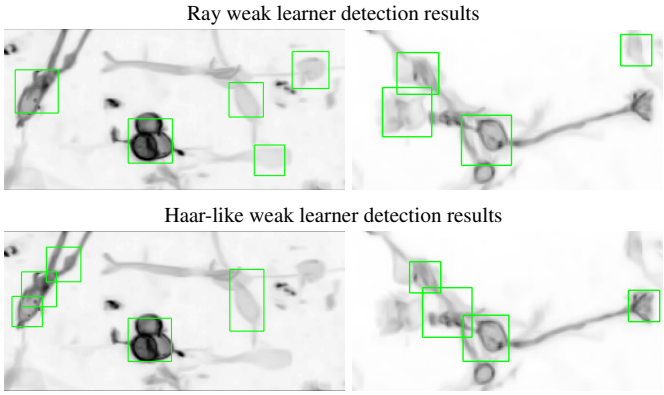

Figure 10. Neuron nuclei detection results. Haar-like features suffer from multiple detections and missed detections *(bottom left)* and growth cones trigger false alarms *(bottom right)*.

Detecting Nuclei in 2-Photon Microscopy Data
This task was driven by our collaboration with neuroscientists to automate the understanding of high throughput microscopy data in an effort to form a better understanding of the development and functionality of the brain. The neuron data set depicted in Fig. 8 contains time-lapse images of fluorescent marked immature neurons migrating in an adult mouse brain. Detecting the nuclei is an important but difficult task; the data contains many distracting objects and shapes including growth cones, neurites, and dead cell matter. The training and validation sets each contained 1,500 positive examples and 1,500 initial negative examples, while more negative examples were sampled from a set of images. The nuclei test set contained 1,500 positive examples and 100,000 negative examples.

The Ray features outperformed Haar and HOG features for the task of detecting neuron nuclei. Figs. 1 and 13(a) show that less learners are needed to train a cascaded detector to similar performance levels, and that for a fixed

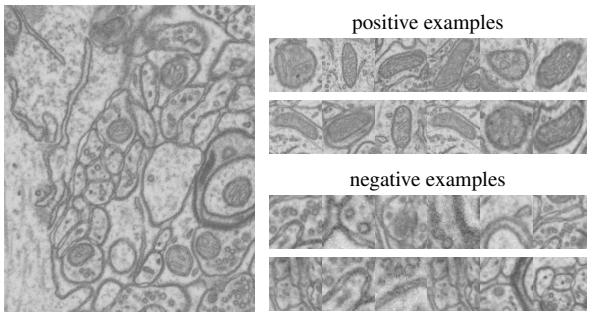

Figure 11. Mitochondria data set. *(left)* An FIBSEM image slice of a mouse brain at the nm scale. *(right)* the task for this data set is to detect mitochondria amongst the clutter of many other blob-shaped objects such as vesicles, synapses, and cross-sections of dendrites and axons.

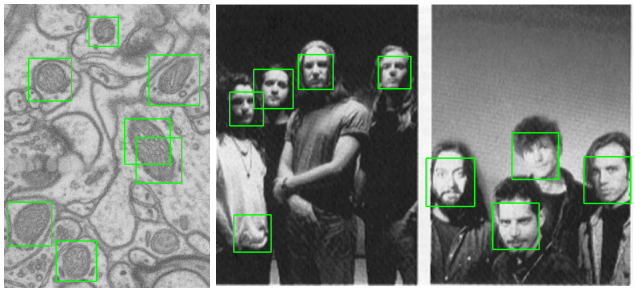

Figure 12. *(left)* Mitochondria detection results using Rays features. *(right)* Face Detection results obtained using a detector trained with Rays features on an image taken from the MIT-CMU database.

number of weak learners, Ray features produce better detection and false positive rates. Fig. 9 shows the classification results obtained on the validation set during training, demonstrating that an AdaBoost classifier trained with Ray features obtains better detection and false alarm rates for a given number of weak learners. Fig. 10 provides detection results for Ray *(top)* and Haar *(bottom)* features.

Detecting Mitochondria in FIBSEM Data

The second task was to detect mitochondria in images of neurons taken using focused ion beam scanning electron microscopy (FIBSEM), as shown in Fig. 11. While the nuclei data set imaged neurons at the μm scale, this set investigates sub-cellular structures inside the neurons at the nm scale. The task of finding mitochondria is made difficult by the presence of many other irregular blob-shaped objects including vesicles, synapses, and cross-sections of dendrites and axons. The training and validation sets each contained 1,200 positive examples and 1,200 initial negative examples, while additional negative examples were sampled from a set of images. The test set contained 1,200 positive examples and 100,000 negative examples.

Rays features also outperform Haar-like and HOG features when applied to mitochondria detection as shown in Fig. 13(b). Detection results are shown in Fig. 12.

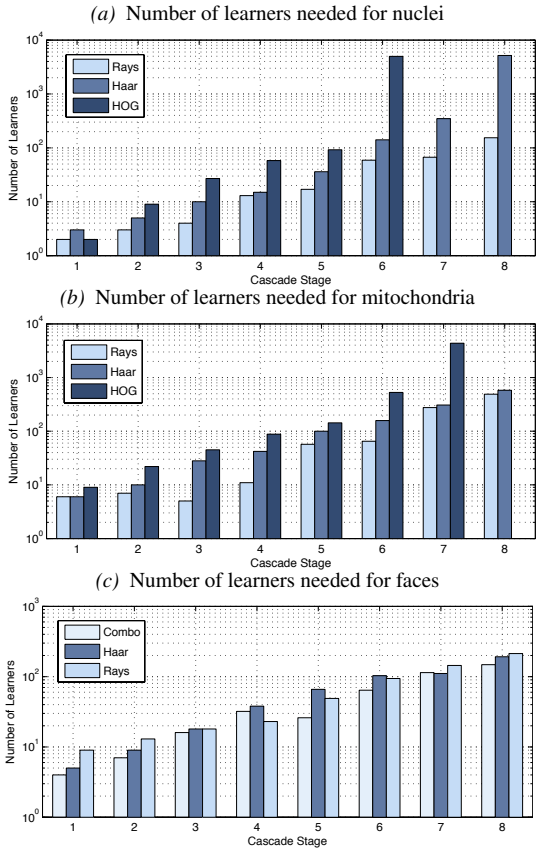

Figure 13. Number of weak learners required per cascade stage. Because the classifier goals are shared among all experiments, we can compare how many learners were required to achieve specific detection D_i and false alarm F_i rates at cascade stage i. Results are shown for the first 8 cascade stages. *(a)* and *(b)* show results for nuclei and mitochondria data. In both cases, Ray features clearly require less weak learners. Haar-like and HOG classifiers need substantially more learners to achieve the same results. However, Ray features cannot outperform Haar-like features on face detection in *(c)* , but combining Ray and Haar-like features results in a more efficient classifier.

Detecting Faces in Photographs

The third task was to detect faces in photographs. Using a web crawler, we collected a new face data set[3] because existing face databases suffered from 1 of 2 problems: (a) they contained too few examples or (b) they contained tightly cropped faces missing the hairline or the chin, which provide essential contours for Rays. The data set contains a training set of 4,000 face examples and 6,000 non-face examples, and a test set of 4,000 face examples and 6,000 non-face examples. The data set also contains 1,653 non-face images used to draw additional negative examples.

Predictably, Ray features do not compete with Haar-like features for the task of face detection. However, Figs. 1 and 13(b) indicate that combining Haar-like features with

[3] Available for download at http://cvlab.epfl.ch/data/.

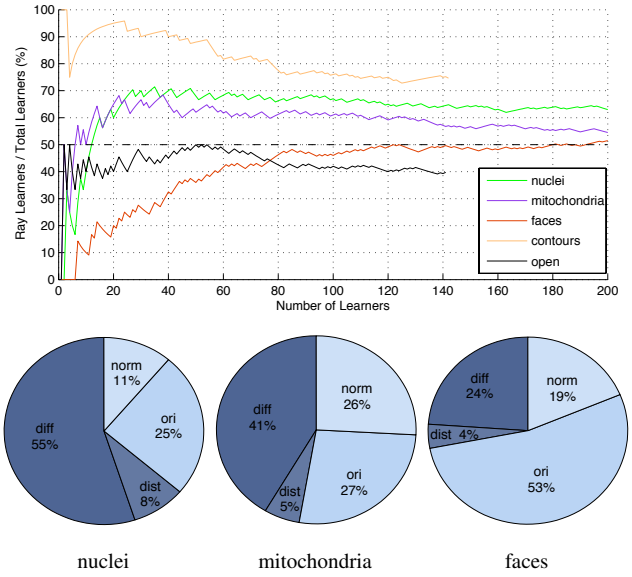

Figure 14. Adaboost learner preferences for different problems. *(top)* Fraction of Total learners selected by AdaBoost which belong to the Rays feature set when combined with Haar-like features. *(bottom)* AdaBoost selection of Rays feature types for various detection problems. See text for details.

Rays features improves the performance of the classifier: less learners are required for a given performance level. Examples faces detected with Ray features appear in Fig. 12.

5. Discussion

To get an insight on how useful Rays features are compared to Haar-like features, we can consider what features AdaBoost selects when given a choice between the two feature types. By looking at several data sets, we can see how the task affects this choice. Fig. 14 *(top)* shows the fraction of total weak learners (out of 73,440 Rays and 95,251 Haar-like) chosen from the Rays feature set for the nuclei, mitochondria, face, contours/noise, closed/open contours data sets. Unsurprisingly, AdaBoost prefers Rays features for problems with irregular shapes: nuclei, mitochondria, and contours; while it prefers Haar-like features for more predictably shaped data: faces and open/closed contours.

It is also interesting understand which Ray features are preferred for a given task. In Fig. 14 *(bottom)* we can see that comparing edge distances ($f_i^{[\text{diff}]}$) are more informative for recognizing the more irregularly shaped nuclei, while edge orientation ($f_i^{[\text{ori}]}$) is more useful when locations of the edges are more predictable, such as for faces.

Acknowledgements

This work was supported by the Brain Mind Institute at EPFL, the European Commission Coordination Action ENINET (contract number LSHM-CT-2005-19063), the Leenaards foundation, the Swiss National Science Foundation, the University of Geneva, and SystemsX.ch the Swiss Initiative in Systems Biology.

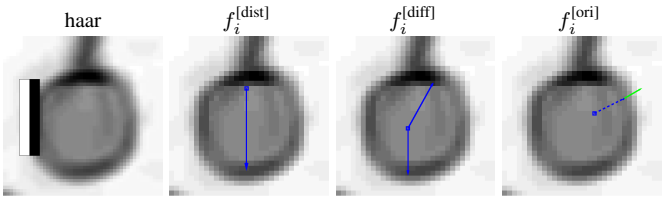

Figure 15. First features selected when AdaBoost is trained on the neuron nuclei data set using Haar and Ray features.

References

[1] Y. Amit and D. Geman. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9(7):1545–1588, 1997.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE T-PAMI*, 24(24):509–522, 2002.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[4] D. Gavrila. Pedestrian detection from a moving vehicle. In *ECCV*, volume 2, pages 37–49, 2000.

[5] D. Geronimo, A. Lopez, D. Ponsa, and A. Sappa. Haar Wavelets and Edge Orientation Histograms for On-Board Pedestrian Detection. In *ICPRIA*, 2007.

[6] C. Grigorescu and N. Petkov. Distance Sets for Shape Filters and Shape Recognition. *IEEE TIP*, 12(10):1274–1286, 2003.

[7] B. Lenseign, P. Brodin, J. Hee Kyoung, T. Cristophe, and A. Genovesio. Support Vector Machines for Automatic Detection of Tuberculosis Bacteria in Confocal Microsocpy Images. In *IEEE Symposium on Biomedical Imaging: From Nano to Macro*, 2007.

[8] R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. In *ICIP*, 2002.

[9] X. Long, W. Cleveland, and Y. Yao. Automatic detection of unstained viable cells in bright field images using a support vector machine with an improved training. *Computers in Biology and Medicine*, 36:339–262, 2006.

[10] T. Mita, K. Toshimitsu, and H. Osamu. Joint Haar-like Features for Face Detection. In *IEEE International Conference on Computer Vision*, Beijing, 2005.

[11] T. Nattkemper, H. Ritter, and W. Schubert. A Neural Classifier Enabling High-Throughput Topological Analysis of Lymphocytes in Tissue Sections. *IEEE TITB*, 5(2):138–149, 2001.

[12] S. Phung and A. Bouzerdoum. Detecting People in Images: An Edge Density Approach. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Honoululu, Hawaii, 2007.

[13] J. Sochman and J. Matas. Waldboost – learning for time constrained sequential detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[14] P. Viola and M. Jones. Robust Real-Time Face Detection. *IJCV*, 57(2):138–153, 2004.

[15] S. Wang and M. Wang. A New Detection Algorithm (NDA) Based on Fuzzy Cellular Neural Networks for White Blood Cell Detection. *IEEE TITB*, 10(1):5–10, 2006.