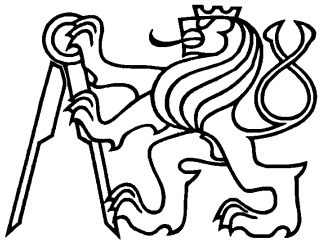
CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING



Diploma thesis

Automatic Malaria Diagnosis through Microscopy Imaging

Vít Špringl

Supervisor: Dr. Ing. Jan Kybic

Prague, 2009

Abstract

Malaria is a an infectious disease which is mainly diagnosed by visual microscopical evaluation of Giemsa stained blood smears. As it poses a serious global health problem, automation of the evaluation process is of high importance. In this work, we attempt to contribute to the problem of automatic malaria diagnosis in several ways. We have developed a graphical user interface for segmentation of red blood cells and for creating a database of red blood cell samples. We also propose a set of features for distinguishing between non-infected red blood cells and cells infected by malaria parasites and evaluate the performance of these features on the set of red blood cells from the created database.

The developed graphical user interface provides all tools necessary for creating a database of red blood cells. It allows a user to execute a segmentation method for a particular blood smear image or for a whole set of images. It enables manual correction of the segmentation results, labeling of segmented objects and saving the results. It can be also used for viewing or editing previously segmented cells stored in the database, it can be easily configured to include new segmentation methods or new labels, and it proved to be a very practical tool. The segmentation technique developed particularly for this task is also described in this work. The method is mainly based on the processing of a thresholded binary image and the watershed transformation is used as a principal method to separate cell compounds. This approach proved to deliver good results on images with various qualitative characteristics resulting in only occasional over-segmented cells.

The main part of this work is devoted to the extraction of features from the red blood cell images that could be used for distinguishing between infected and non-infected red blood cells. We propose a set of features based on shape, intensity, and texture and evaluate the performance of these features on the red blood cell samples from the created database using receiver operating characteristics. The results have shown that some of the features could be successfully used for malaria detection.

Keywords:

Malaria; Plasmodium; red blood cell; image processing; microscope image analysis; segmentation; feature extraction; Matlab program

Abstrakt

Malárie je závažné infekční onemocnění, které je diagnostikováno mikroskopickou analýzou Giemsou barvených krevních roztěrů. Protože toto onemocnění představuje závažný celosvětový problém, na automatizaci celého vyhodnocovacího procesu se klade velký důraz. Tato práce popisuje grafické rozhraní, které bylo vyvinuto pro účely segmentace obrazů krevních roztěrů a pro vytvoření databáze červených krvinek. Tato databáze je následně použita pro vyhodnocení sady příznaků, které byly navrženy pro detekci červených krvinek napadených malarickými parazity.

Navržené grafické rozhraní poskytuje veškeré nástroje potřebné k vytvoření databáze červených krvinek. Program umožňuje segmentovat vybraný vstupní obraz nebo celou sadu obrazů pomocí navrženého segmentačního algoritmu a dále umožňuje manuální korekci výsledků segmentace, přiřazení třídy daným objektům a uložení výsledků. Program lze rovněž použít k prohlížení a editaci již segmentovaných obrazů červených krvinek uložených v databázi. Nové segmentační metody a třídy mohou být do programu snadno přidány editací příslušných souborů. Segmentační metoda, která je využívána grafickým rozhraním, je rovněž popsána v této práci. Metoda je založena zejména na zpracování binárního obrazu získaného prahováním, přičemž pro oddělení překrývajících se červených krvinek se využívá watershed transformace.

Hlavní část této práce je věnována extrakci příznaků z obrazů červených krvinek, které by mohly být použity pro rozpoznávání a detekci infikovaných červených krvinek. Popsány jsou příznaky vycházející z tvaru objektu, jasu a textury. Jednotlivé příznaky jsou vyhodnoceny na obrazech červených krvinek z vytvořené databáze s využitím ROC křivek. Výsledky ukazují, že některé příznaky by bylo možné s úspěchem použít pro detekci malárie.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a uvedl všechny použité prameny.

V Praze dne 20. ledna 2009

Vít Špringl

Acknowledgements

I would like to express my thanks to Dr. Ing. Jan Kybic for his recommendations and support with this thesis and for being my supervisor.

I would also like to thank Juan David García-Arteaga for his valuable suggestions and advice.

Contents

1 Introduction

Malaria is a serious global disease and a leading cause of morbidity and mortality in tropical and sub-tropical countries. It affects between 350 and 500 million people and causes more than 1 million deaths every year [30]. Yet, malaria is both preventable and curable. Rapid and accurate diagnosis which enables prompt treatment is an essential requirement to control the disease [31].

The most widely used technique for determining the development stage of the malaria disease is visual microscopical evaluation of *Giemsa* stained blood smears. This process consists of manually counting the infected red blood cells against the number of red blood cells in a slide. The manual analysis of slides is, however, time-consuming, laborious, and requires a trained operator [40,41]. Moreover, the accuracy of the final diagnosis ultimately depends on the skill and experience of the technician and the time spent studying each slide [39] and it has been observed that the agreement rates among the clinical experts for the diagnosis are surprisingly low [42]. In this context, the development of a mechanism that automates the process of evaluation, quantification and classification in thin blood slides becomes a high priority and the aim of this work was to contribute to improvement upon malaria microscopy diagnosis by removing the reliance on the performance of a human operator for diagnostic accuracy.

A number of methods have been proposed for automatic parasite detection in Giemsa stained blood films based on different approaches. These approaches include pixel-based parasite detection [17,21], detection based on morphological processing of segmented parasites [2,3], or detection by extracting image features from the segmented cells [4]. These methods are summarized in section 5. In this work, evaluation of malaria is based on the last approach.

The problem with many published papers is that the exact definitions of methods, features and their parameters used for parasite detection are not presented in sufficient detail to allow the reader to repeat the experiment. We already encountered this sort of problem in our previous work [1]. In this work, we propose a set of features and evaluate the performance for a general problem of distinguishing between infected and non-infected red blood cells. Some of these features have already been used in other works but some of them may be new for the problem of malaria parasites detection. Exact definition of these features is provided, including description of the parameters controlling the generation of the transformed images and description of the preprocessing steps performed. Individual sets of features are evaluated on a created dataset of red blood cell samples using ROC curves for different parameters controlling the feature extraction. Evaluation is followed by a discussion on the effects of different preprocessing techniques and possible utilization of these features for more specific problems of distinguishing between different types of malaria parasites.

In order to create a database of red blood cell samples, a method for segmentation of red blood cells in the input blood smear images have been proposed and a graphical user interface have been devised for manual correction of the segmentation results and for

manipulation with the samples comprising the database. The segmentation method is partially based on a previously developed technique, which is described in [1], but many modifications have been made to improve reliability and to conform to new requirements.

This report is organized in the following way. In section 3, theoretical background about malaria and blood smears acquisition is given. In section 4, the input set of images and the programming environment, in which all the methods are implemented, are briefly described. Section 5 gives an overview of the state-of-the-art methods proposed in other works. The segmentation method is described in section 6 and details on the developed GUI are given in section 7. Section 8 briefly describes the structure of the created red blood cell database. In section 9, the proposed set of features is described and the results of their evaluation are presented. Finally, section 10 draws the conclusions. The appendices include the plots of the estimated probability density functions and ROC curves for the evaluated features and brief description of all files that are distributed as part of this work.

2 Objective of the Thesis

The objective of this diploma work was to create a database of red blood cell from available Giemsa stained blood smear images containing cells infected by Plasmodium parasites and propose and evaluate a set of features that could be used for detection of Malaria. To automate the process of database creation, a segmentation method was to be proposed, which would determine the regions in the original image corresponding to the individual red blood cells and separate overlapped and occluded cells. The second task was to develop a tool with graphical user interface for manual correction of the segmented objects, labeling, and editing the cells in the database. The goal was to create a program providing all necessary tools needed for creation of and manipulation with the red blood cell database and that could be easily configured to be possibly used in future works. The individual cell images comprising the database should be stored in a transparent and easily accessible way so that they can be readily retrieved by anyone willing to perform malaria detection experiments. The aim was also to preserve in the database as much as possible information contained in the original blood smear images.

The purpose of the feature extraction and evaluation part was to provide a general guidance for a reader interested in designing a classifier for detection and evaluation of Malaria infection. The aim was to present a detailed description of a set of features that could possibly be used in Malaria diagnosis, including exact definitions of the image transformations and measurements, description of preprocessing methods, evaluation of feature performance on a general problem of distinguishing between non-infected and infected red blood cells, and discussion on the suitability of the features and their sensitivity to various deficiencies among the input images, such as noise, or color and contrast variance.

3 Background

3.1 Malaria

Malaria is caused by protozoan parasites of the genus *Plasmodium*. There are four species of *Plasmodium* that infect man and result in four kinds of malarial fever: *P. falciparum*, *P. vivax*, *P. ovale*, and *P. malaria*e [33]. *P. vivax* shows the widest distribution and is characterized by reappearances of symptoms after a latent period of up to five years. With the similar characteristics, *P. ovale* appears mainly in tropical Africa. *P. falciparum* is most common in tropical and subtropical areas. It causes the most dangerous and malignant form of malaria without relapses and contributes to the majority of deaths associated with the disease [32]. *P. malariae* is also widely distributed but much less than *P. vivax* or *P. falciparum*.

There are three phases of development in the life cycle of most species of plasmodia [33]: *exo-erythrocytic stages* in the tissues, usually the liver; *erythrocytic schizogony* (i.e. protozoan asexual reproduction) in the erythrocytes; and *the sexual process*, beginning with the development of gametocytes in the host and continuing with the development in the mosquito.

When an infected mosquito bites humans, several hundreds *sporozoites* (the protozoan cells that develop in the mosquito's salivary gland and infect new hosts) may be injected directly into the blood stream, where they remain for about 30 min and then disappear. Many are destroyed by the immune system cells, but some enter the cells in the liver. Here they multiply rapidly by a process referred to as *exo-erythrocytic schizogony*. When schizogony is completed, the cells produced by asexual reproduction in the liver termed *merozoites* are released and invade the erythrocytes. In *Plasmodium vivax* and *P. ovale*, some injected sporozoites may differentiate into stages termed *hypnozoites* which may remain dormant in the liver cells for some time before undergoing schizogony causing relapse of the disease.

When the released merozoites enter erythrocytes, the erythrocytic cycle begins. This process is referred to as e*rythrocytic schizogony*. Within an erythrocyte, the parasite is first seen microscopically as a minute speck of chromatin surrounded by scanty protoplasm. The plasmodium gradually becomes ring-shaped and is known as ring or immature *trophozoite* (Fig.1a). It grows at the expense of the erythrocyte and assumes a form differing widely with the species but usually exhibiting active pseudopodia (i.e. projections of the nuclei). Pigment granules appear early in the growth phase and the parasite is known as a mature trophozoite (Fig.1c). As the nucleus begins to divide, the parasite is known as a *schizont* (Fig.1d-f). Dividing nucleus tends to take up peripheral positions and a small portion of cytoplasm gathers around each. The infected erythrocyte ruptures and releases a number of merozoites which attack new corpuscles and the cycle of erythrocytic schizogony is repeated. The infection about this time enters the phase in which parasites can be detected in blood smears.

Some merozoites on entering red blood cells become sexual gametocytes, instead of asexual schizonts. When gametes are ingested by a mosquito, the cells rapidly undergo gamete production. This is the third phase of development in the life of plasmodium, the sexual process of reproduction in a mosquito.

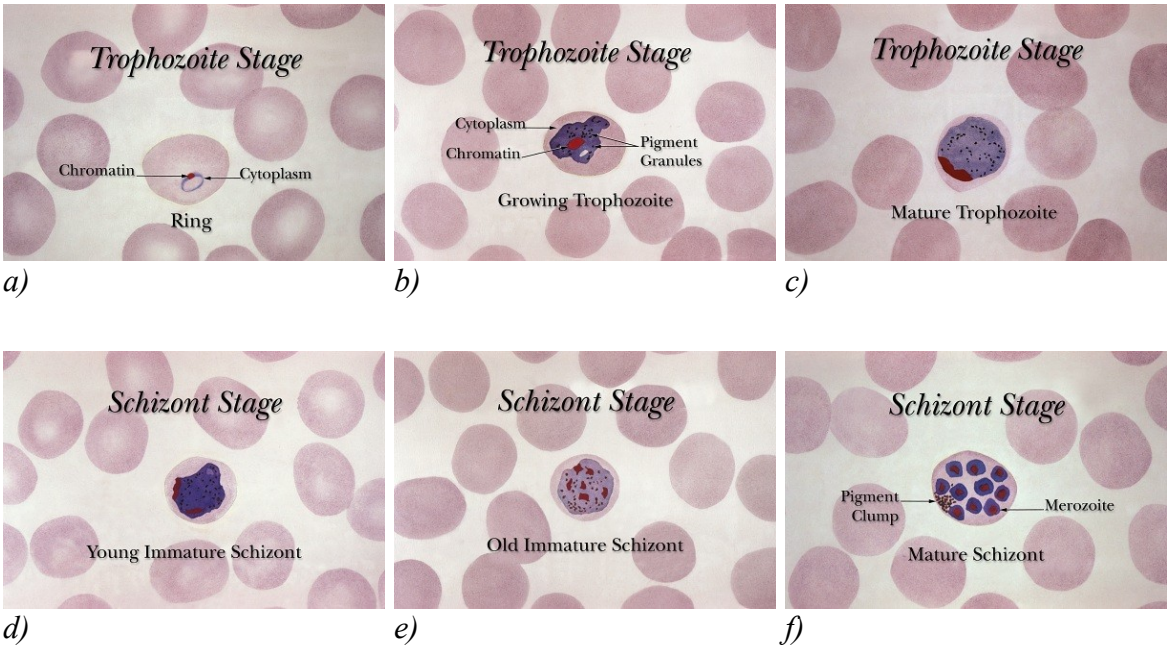


Fig. 1: Development stages of the Plasmodium parasite. Image courtesy of CDC [35]

3.2 Giemsa Stain

Giemsa stain is used to differentiate nuclear and cytoplasmatic morphology of platelets, red blood cells, white blood cells and parasites [34]. Giemsa staining solution stains up nucleic acids and, therefore, parasites, white blood cells, and platelets, which contain DNA, are highlighted in a dark purple color. Red blood cells are usually colored in slight pink colors.

3.3 Peripheral Blood Smears

Peripheral blood smears or blood films are microscopic slides prepared from a blood sample that allow microscopical examination of blood cells. Blood smears are typically used for investigation of hematological disorders and for detection of parasites, such as the Plasmodium. Two sorts of blood smears are traditionally used [36]. Thin blood smears allow better species identification, because the appearance of the parasites is better preserved in this preparation. Thick blood smears allow screening of a larger volume of blood and, therefore, they can give more than a ten-fold increase in sensitivity over thin

films. However, the appearance of the parasite is more distorted and, therefore, distinguishing between the different species can be more difficult.

In principle, blood films are prepared by placing a drop of blood on one end or into the center of a slide and spread with the corner of another slide or a swab stick to cover an oval area along the slide. The aim is to get a region where the cells are sufficiently spread to be counted and differentiated. The well spread part of the blood smear, specifying the working area for microscopic analysis, is defined as a zone that starts on the body film side when red blood cells stop overlapping and finishes on the feather edge side when red blood cells start to lose their clear central zone [37]. The smear is then thoroughly dried in an incubator at 37ºC for around one hour. The dry film can be subsequently stained using Giemsa dilution.

For malaria diagnosis, blood films should be prepared as soon as possible after blood samples are taken. Such films adhere better to the slides, leave a clearer background after lysis, and parasite and red cell changes are minimal [36]. This is a big problem in many laboratories in developed countries, because the delay left between taking the blood sample and making the blood films is too long. Further development of the sexual stages may occur even within 20 minutes under the right conditions and the male gametes released into the plasma may be mistaken for other organisms, such as Borrelia. If infected blood is left at warmer temperatures, schizonts will rupture and red cells may be invaded by released merozoites. These can mistakenly give the appearance of other forms of Plasmodium parasites. Moreover, parasite and red blood cell morphology can be seriously affected if anticoagulants have to be used and blood has been in anticoagulant for too long time [38].

Microscopic examination of blood films is the most efficient and reliable malaria diagnosis technique and is very sensitive and highly specific, because each of the four major parasite species has distinguishing characteristics [39]. It allows differentiating between species, quantification of parasitemia, and observation of asexual stages of the parasite. Moreover, low material costs make the marginal costs of test very low [40].

4 Materials and Methods

4.1 Blood Smear Images

Images of Giemsa stained blood smears were selected from the Public Health Image Library [35] and they have the following common characteristics.

- Images are available in different magnifications and sizes. The images are available in TIFF format with the resolution of 2 to 3 megapixels
- Digital images are obtained by scanning and, therefore, contain apart of the noise and artifact from the sample and from the microscope light also noise from the chemical development process or from the scanner.
- Images exhibit high variability in color tone, intensity, contrast, and illumination. The overall color tone varies significantly from grayish, blue, purple, and pink to yellowish and it may even change from the center of the image to its borders (Fig.2). Some images have very low contrast (Fig.3a) while some images exhibit hight contrast between infected and non-infected cells (Fig.3f). Many images suffer from irregular illumination.
- The overall shape and appearance of the cells may also vary substantially among the slides. Some cells lack their clear central parts (Fig.3b) and, in some images, cells may assume shapes that differ from the usual circular shape (Fig.3c). Moreover, red blood cells are often overlapping and may form big clusters (Fig.3d). Occasionally, blurring and various artifacts may also appear (Fig.3e).

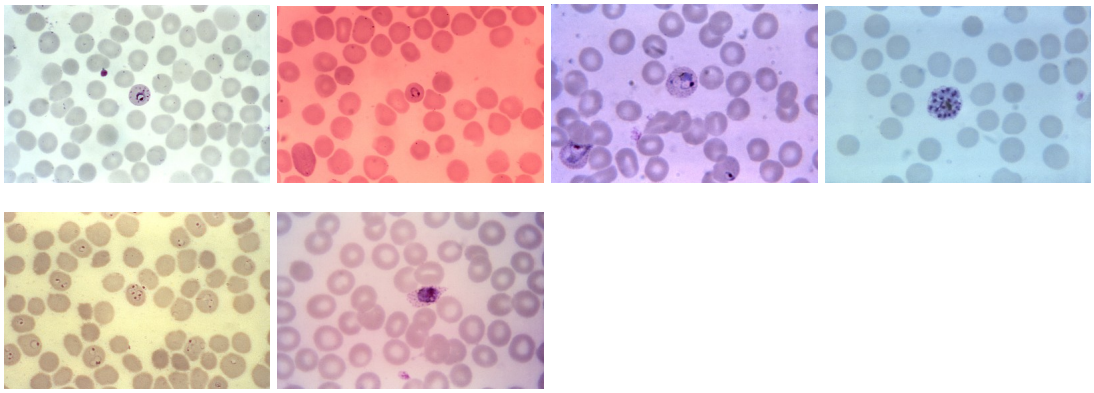


Fig. 2: Samples of available stained blood smear images showing differences in color tone and illumination. Image courtesy of CDC/ Dr. Mae Melvin, Steven Glenn [35]

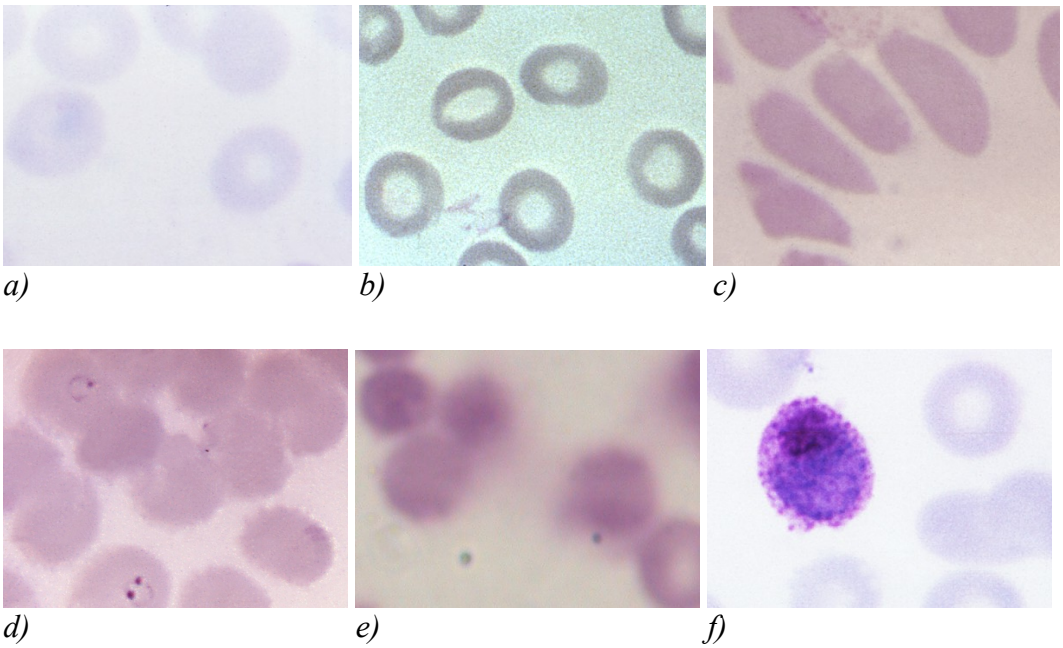


Fig. 3: Cropped samples of available blood smear images showing different qualitative characteristics of the input images. Image courtesy of CDC / Dr. Mae Melvin, Steven Glenn [35]

4.2 Matlab Programming Language

The entire project, including all functions, segmentation GUI, image database files, and feature evaluation scripts, have been implemented using Matlab programming environment version 7.7.0. Although we cannot guarantee that all functions will work properly with any older version of Matlab, most of the functions, and especially the GUI, were tested in older versions of Matlab and changes were made where necessary to ensure backward compatibility.

Matlab is a high-performance language for technical computing, which integrates computation, visualization, and programming in an easy-to-use environment. It is an interactive system using an array that does not require dimensioning as the basic data element. Typical uses include math and computation, algorithm development, data acquisition, analysis and visualization, modeling and simulation, scientific and engineering graphics, and application development including graphical user interface building. It removes the need of programming many routine tasks for numerical computing and allows easy and quick displaying of results both in numerical form as well as in the form of 2D or 3D graphs. The open-architecture of Matlab allows programmers to incorporate their own area specific set of functions implemented in separate m-files into Matlab, so that they can

be easily used by other functions and scripts written in Matlab. Information about digital image processing using Matlab and about programming graphics and GUIs with Matlab can be found, for example, in [43] and in [44], respectively.

The following toolboxes are used and required in order to run all functions and scripts in the project:

- Image Processing Toolbox
- Statistics Toolbox
- Optimization Toolbox

The Image Processing toolbox is essential as it is used by most of the functions and scripts and it is the only toolbox required for running the segmentation method and the GUI. The Statistics Toolbox is used in feature extraction functions and the Optimization Toolbox is not strictly required as it is only used for computation of the shape measurements.

5 The State of the Art

Automatic malaria diagnosis based on Giemsa stained blood smear images have been addressed in several works using different approaches.

In the work of Ross et al [4], an image processing techniques is described that is used to identify erythrocytes and possible parasites present on microscopic slides. The algorithm consists of preprocessing of the image, image analysis, image segmentation, generation of features, and classification of an erythrocyte as infected with malaria or not.

The preprocessing and the image analysis parts follow, to a certain extent, the works of Di Ruberto [2, 3]. The preprocessing of the image includes filtration by a 5×5 median filter, followed by a morphological area closing filter. Only the green component of the true color original image is used for image analysis and segmentation.

The image analysis step includes calculation of the size and eccentricity of the erythrocytes and differentiating free-standing cells from the overlapping ones. The size of erythrocytes is determined via pattern spectrum which is calculated using granulometry. Granulometry is computed from the difference in morphological openings using increasing sizes of structuring elements. Free-standing erythrocytes are differentiated from overlapping cells by their area.

In the image segmentation step, potential parasites and erythrocytes are identified and segmented from the background. The segmentation of the erythrocytes is accomplished by the means of thresholding of the green image component. The holes in the resulting thresholded binary mask of erythrocytes are removed using morphological opening filter. To segment potential parasites, local and global thresholding levels are used and the resulting binary images are combined to obtain the parasite marker image. To separate clusters of cells, a series of morphological operations is applied.

Two sets of features are proposed to separate classes. The first set is based on image characteristics that have been used previously in biological cell classifiers. These include geometric features, such as shape and size, color attributes derived from the red, green, blue, hue and saturation components (includes measures such as peak intensity, average intensity, skewness, kurtosis, and entropy of the component histograms), and gray-level texture features.

The second set of features utilizes measures of parasites and infected erythrocytes morphology that are commonly used by technicians for manual microscopic diagnosis. These features include: the relative size and the relative eccentricity of infected erythrocytes, smoothness of the cell margin, the relative color of infected erythrocytes, texture information, the number of parasites per erythrocyte, the number of chromatin dots per parasite, morphology of the rings, and others.

An erythrocyte is classified in a two-stage process. First, an infection is classified as positive or negative and, in case it is classified as positive, the species is assigned at the second node. Backpropagation feedforward neural networks are used for classification.

Besides listing the features, the paper gives no details on which of the generated features were exactly used for the classification and includes no definitions or further specifications of the features.

In the works of Di Ruberto et al. [2, 3], objects have been detected by means of an automatic thresholding on single components of the RGB and HSV histograms based on a morphological approach. The articles describe morphological methods for both cell image segmentation and parasites detection. The proposed technique uses granulometries to evaluate the size of the red cells and the nuclei of parasites and regional maxima to detect the nuclei of parasites. Morphological techniques, such as thinning, gradient, reconstruction by dilation, and morphological filters are also utilized for pre- or post-processing of the images.

Red blood cell segmentation is performed by thresholding the green component image with non-uniform illumination correction using fixed paraboloid model of the illumination. Morphological area-opening filters are used to remove items smaller than a red blood cell and to fill the 'holes' left after thresholding. For the radius of the structuring element to be correctly set, the smallest size of the red cells is estimated from the size distribution based on the granulometric analysis. The morphological gradient is applied on the result to obtain a binary image which is used as a marker image in the watershed-based segmentation to find the contours of the red blood cells. Composite cells which are distinguished based on the roundness ratio are separated by applying a morphological opening filter.

The product of the binary thresholded H and S images is used as the marker image for the parasites and white blood cells detection. Morphological area closing on both the H and S components and the regional maxima is used to pre-process the images and the granulometric analysis is applied in order to evaluate the red cells sizes and to set accordingly the radius of the structuring element. The average gray levels of the nuclei marked by the intersection of the regional maxima on the H and S images are used as threshold values to detect the parasites and white blood cells in the H and S images.

White blood cells are isolated from the parasites by means of a morphological erosion on the product of the binary thresholded H and S images. The white cells marked by this erosion are then reconstructed by dilation.

Parasites are classified into four classes: immature trophozoites, mature trophozoites, gametocytes and schizonts. The schizonts are identified from the thresholded H and S marker image as areas with high clustering. To measure the separation of the objects in the image plane, Hausdorff distance [45] is used. All the objects with distance smaller than the average size of the red cells are considered to be schizonts. The remaining objects identify nuclei of parasites and are further classified as immature and mature trophozoites and gameotcytes by analyzing the shape of the parasite. The presented method uses endpoints of digital skeletons generated by thinning algorithms to describe the shape and distinguish between the species of the parasites. The details of the classification method used were not published.

In [3], some more details are given on the background of the granulometric analysis and regional extrema and a second classification method is presented in addition to the one

described already in [2]. The second approach analyses the area around the nucleus of the parasite looking for the presence or the absence of small spots of chromatin around it and compares it with a sample image object set based on color histogram similarity. Color histogram ensures the invariance to translation and rotation of objects, partially occlusions, and normalizing the histogram with respect to the object area leads to scale invariance. The histograms are computed on the infected red cell area, which are not nuclei of the parasites. The HSV color space is used to compute the histograms. This method, however, strongly depends on the choice of the reference colors and color space representation and requires sample objects which are used as prototypes. The system assumes stable color tone and intensity among the stained images, defined illumination conditions, and known noise characteristic.

An automated image analysis-based software "MalariaCount" for parasitemia determination, i.e. for quantitative evaluation of the level of parasites in the blood, has been described in [46]. The presented system is based on the detection of edges representing cell and parasite boundaries.

The described technique includes a preprocessing step, edge detection step, edge linking, clump splitting, and parasite detection. The preprocessing of the image, which involves the enhancement of the image contrast via adaptive histogram equalization, is followed by edge detection, where a pixel is determined to belong to the boundary edge of the red blood cells if a defined edge correlation coefficient exceeds an empirically determined threshold. The resultant edge contours are linked together through their terminal points to form closed boundaries. The terminal points are identified using 20 different 3×3 masks. In order to split the red blood cell clumps, the concavity pixels are detected and concavity-based rules are applied to generate the split lines. Parasites are detected as regions with large edge response magnitude inside the red blood cells. Parasites are neither classified nor any other analysis is performed on the objects detected within the red blood cells.

The system requires well-stained and well-separated cells in order to provide accurate result. Moreover, artifacts, 'holes' inside red blood cells and noise can lead to a false interpretation of a red blood cell. The program is not intended for studies involving patient samples.

The paper by Díaz et al. [17] evaluates a color segmentation technique for separation of pixels into three different classes: parasite, red blood cell and background, based on standard supervised classification algorithms. Four different supervised classification techniques – KNN, Naive Bayes, SVM and Neural network – are evaluated on different color spaces – RGB, normalized RGB, HSV and YCbCr.

Before applying the classification process, the luminance differences in the original images were corrected using a local adaptive low pass filter defined for a window size of the larger image feature, in this case a typical red blood cell size. The images were subsequently represented in four different color spaces.

In order to separate pixels into one of the three classes, a set of training samples was manually extracted by an expert and each pixel of the training sample was labeled accordingly. Using this set of sample pixels, a classification model was trained for each of

the tested color spaces (RGB, normalized RGB, HSV an YCbCr). The classified color space was then used as a look-up table. Each classification model was tuned independently for its own particular set of parameters and all experiments were performed using a 500 elements training data set.

Two kinds of evaluation were performed. In the *pixel-wise evaluation*, each pixel is classified as parasite, red blood cell or background. The best overall performance was accomplished by the combination of a KNN classifier and YCbCr color space. The results show that the performance is generally significantly lower for the parasite classification than for the red blood cell classification. It is concluded in the paper that the complex mix of colors present in the parasites makes it difficult to discriminate individual pixels using only color information. In the *interest-object wise evaluation*, the same classification process is performed as the first step. After the application of the classification method, a basic filtering process is performed to keep only relevant objects in the image. During this process, small and large regions identified as flaws are removed and near unconnected segments are evaluated for their relevance to a given parasite and if found relevant, they are considered as a unique object. In this case, the test set is composed of images where interest-objects, i.e. erythrocytes and parasites, are labeled, whereas in the first case, the test set was composed of labeled pixels. The achieved performance of the interest-object wise evaluation was much better than the one achieved at the level of pixel classification, with the best overall performance accomplished by the combination of a KNN classifier and normalized RGB color space.

The article presents a simple method for red blood cell and parasite detection with no classification of parasites. The approach is based on a classification process that finds boundaries that optimally separate a given color space. No details on the filtering process performed to separate the relevant objects of interest are given. The system assumes constant color tone in the input images, since only luminance differences are corrected.

The paper by Tek et al. [21] presents a method to detect malaria parasites using a Bayesian pixel classifier to first separate stained and non-stained pixels and a distance weighted K-nearest neighbor classifier to further classify the stained pixels as parasites or non-parasites. The second classification is performed using four selected features: color histogram, Hu moments, shape measurements, and color auto correlogram, which are all rotation and scale invariant.

Before applying the classification process, color of the input images is normalized to decrease the effect of different light sources or sensor characteristics. The color normalization is performed using an adapted gray world normalization method [22] based on the diagonal model of illumination in which an image of unknown illumination can be simply transformed to known illuminant space by multiplying pixel values with a diagonal matrix.

To classify pixels as stained or non-stained, a Bayesian classifier using an RGB color vector as a feature was utilized. The class conditional probability density functions are estimated using a non-parametric method based on histograms. A training set of images with all the stained objects manually labeled was formed to calculate the probability density functions.

The stained structures identified in the first classification step may contain, in addition to parasites, also other components, such as white blood cells, platelets or staining artifacts. In order to identify the structures among the stained pixels, infinite morphological reconstruction [10] is applied using the stained pixels as markers and the negative gray level image to approximate the cell region which includes the stained group. This process merges some stained pixel groups belonging to the same regions and allows some of the non-parasites to be eliminated by comparing to the estimated average cell size and location (background, foreground). The labeled stained pixel groups are subsequently used for the computation of the features.

To classify the stained pixels as parasite or non-parasite, a distance weighted K-nearest neighbor classifier was utilized using four selected features – color histogram, Hu moments, relative shape measurements vector, and color auto correlogram. The relative shape measurements vector is formed of simple measurements representing the object shape.

According to the results of the study, the most successful feature to classify the stained objects as parasite/non-parasite was the combination of correlogram, Hu moments and relative shape measurements.

6 Red Blood Cell Segmentation

To extract features of an object, we need to separate the objects of interest from the background and from each other and define the zone of measurement, i.e. the region where to measure the characteristics of the object. The objects of interest are in our case red blood cells which are either infected or not by the plasmodium parasite. The zone of measurement is the area of the whole single cell. This area can be described and stored in form of binary mask or as a contour of the object.

The segmentation technique, which is later used by the GUI, is based on an algorithm developed and described in our previous work [1]. However, some changes have been made to improve the performance of the algorithm on larger variety of input images and to reflect certain changes in requirements, mainly that we are now more interested in obtaining correct shape of the segmented cell than the correct number of red cells in the image and that we rather allow certain number of over-segmented cells, which can be rather easily merged in the GUI, than under-segmented cell clusters. The algorithm also provides the output data in a defined format that can be read by the GUI.

The segmentation of the input image is a crucial step in almost all image analysis tasks and it is often also one of the most difficult ones [5]. The segmentation of the red blood cells can be performed via thresholding with an automatically estimated threshold followed by mask processing, which may include separation of the overlapping cells, removing artifacts or objects too small to be red cells, or correcting shapes of the segmented cells [2,3,4,5]. An algorithm based on this approach have been developed and used in this work.

6.1 The Initial Algorithm

The aim of the study [1] was to segment red blood cells in Giemsa stained blood slides and count the number of infected cells versus the total number of red cells in the image in order to evaluate malaria.

The presented segmentation technique consists of several steps. The input RGB image is first converted to the gray level representation by using only the green channel of the original image. The noise in the image is then smoothed by a median filter using 3x3 window.

The average radius of the red blood cells in the image is an important parameter for many subsequent operations and is evaluated using a voting technique which was introduced in [6]. This technique detects edge pixels in the image using Laplacian of Gaussian method and moves each edge pixel in the direction of the image gradient at its location for different values of the distance parameter *r*. Since the red blood cells are approximately circular in shape, the translated edge pixels will form clusters in the centers of the cells for a certain value of the distance parameter *r*. The distance which produces the highest local maxima in the image with translated edge pixels is considered to be the average radius of the cells in the image.

The pre-processing part continues by correction of non-uniform illumination in the gray-scale image. Background illumination is obtained by applying gray-scale morphological closing using a disk-shape structuring element of size $1.5 \cdot R_A$, where R_A is the estimated average cell radius. Resulting image filtered by a Gaussian filter is then used as a multiplicative error coefficient to compute the approximation of the undegraded image with uniform illumination [7].

Red blood cell segmentation is performed using marker-controlled watershed transformation based on the image gradient [7]. Markers are computed as a combination of the binary mask of the red blood cells and centers of the cells which are computed using a similar algorithm that was utilized for evaluation of the average cell radius. The binary mask is obtained by thresholding the gray-scale image with an automatically estimated threshold using Otsu's method [8].

Although this algorithm performed well on the images tested in [1], certain issues appeared when applying it on a new set of blood smear images with higher resolution. The noise in the images was more pronounced and red blood cell margins showed lower gradient magnitude (the contours of the cells appeared to be blurrier). This caused the edge detector using the Laplacian of Gaussian method to fail and mark many pixels of noise while omitting some pixels of cell contours (Fig.4a). As a result, the average cell radius found by this method was, in some cases, far from the real cell radius, typically a very small number. The wrong value of the cell radius resulted in a poor performance of the succeeding methods.

Another issue appeared when using the marker-controlled watershed transformation based on image gradient. The markers are very important for correct separation of overlapping cells as well as for obtaining the correct mask of a cell. If a marker is placed outside the boundary of any cell, a mask that does not correspond to any object in the image may be produced. On the other hand, if a marker is not present within an area of an object that is to be separated from the others or from the background, the mask of the object as a result of the watershed transformation may be contorted or even not present at all. The markers are derived from the local maxima in the 3-dimensional accumulator matrix $[x,y,r]$ representing the $[x,y]$ locations of the edge pixels moved in the direction of their gradient by the distance of r. It proved to be a non-trivial task to adjust the criteria for the marker selection on the new image set so that the markers represent only the real centers of the red blood cells and as many as possible cell centers are included in the marker set. A marker can be wrongly evaluated for different reasons. It can be omitted, for example, when a cell contour is blurred and produces less edge points or when a cell is elongated and has a non-circular shape.

In some images, red blood cells have ring-like shapes with center intensity very close to the intensity of the background (Fig.3c). Thresholding of such images may produce ring-shaped masks of red blood cells with 'holes' inside and these images also usually have relatively high magnitude of the gradient within the cell which may be comparable with the gradient magnitude produced at the contour of the cell. This caused, in certain cases, additional inaccuracies in the shapes of the generated masks.

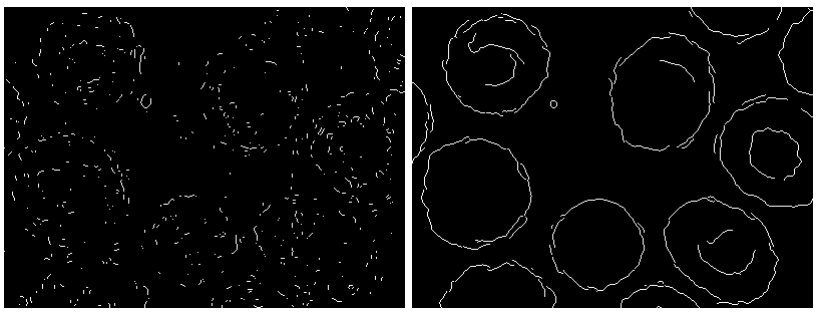
6.2 New Algorithm

Although the issues mentioned in the previous section often caused minor errors when evaluating the total number of cells in the image in [1], manual corrections of the shapes of the masks were often required. To make the results more easily manually correctable when creating a red blood cell database for classification purposes using the designed GUI, a modified segmentation method was implemented. The aim was to develop a more robust algorithm with better performance on new images which produces more predictable results with correct contour shapes.

The new algorithm uses the same functions in the preprocessing stage with only minor modifications. The input RGB image is first converted to a gray-scale image by using only the green channel of the RGB image. The gray-scale image is then filtered by median filter and after computing the average cell radius, the illumination correction is performed. The filtered gray-scale image with corrected illumination is then converted to a binary image by thresholding the image with an automatically estimated threshold using the same method as is described in [1].

The problem with edge detection described in the previous section that caused the function evaluating the average cell radius to provide misleading results was corrected by changing the size of the window of the median filter to 8×8 and by using the Canny method for edge detection [7,9]. Canny edge detector proved to produce more connected edge lines at the contours of red blood cells than the Laplacian of Gaussian method while marking only a few noise pixels as edges (Fig.4).

For the problems mentioned in the previous section, the marker-controlled watershed transformation based on image gradient is no longer used. Instead, a rather simple and straightforward approach was adopted producing more predictable results with masks better representing the cell shape. This approach uses only the binary mask image. The overlapping cells are separated using watershed transformation based on distance transformation of the binary image.



a) *b)*

Fig. 4: Comparison of edge detection methods. a) Laplacian of Gaussian method, b) Canny method.

6.2.1 Filling Holes in Red Blood Cells

As mentioned in section 6.1, red blood cells in some blood smear images have ring-like shapes and the corresponding binary mask obtained by thresholding of such an image may contain holes in the centers of the cells. Such holes have to be filled in order to obtain correct masks of the red blood cells and to allow the subsequent methods to work properly.

A hole in a binary image is a set of pixels of background surrounded and enclosed by pixels of foreground. Since in all images the background is connected to the edges of the image, we can use a flood-fill operation starting with the border pixels of the background to fill the background area connected to the edges and identify the holes in the image as background pixels that cannot be reached by such operation [10]. Such holes can be filled by simply inverting the values of the pixels that were not reached by the flood-fill operation marking them as foreground.

Although this operation will work in most cases, the disadvantage of such approach is that not all such holes in the image are necessarily also the holes in the centers of red blood cells to be filled. A region of background pixels may, for example, be surrounded by three or more connected cells and thus also separated from the rest of the background and identified as a hole. Markers of the cell centers that are together with the estimated average cell radius computed by the function `getCenters` can be successfully used in the task of distinguishing the holes in the image that are located in the centers of red blood cells. The holes in the cells to be filled can be identified by computing the intersection of each set of pixels which was marked as a hole by the filling operation with the set of marker pixels of cell centers. If such intersection results in an empty set, the area is not considered as a hole to be filled.

Two more operations are performed before computing the intersection of the two sets. Marker image contains only single pixels representing the estimated locations of cell centers. Since the holes to be filled are not necessarily located in the centers of the cells, each cell center marker should cover a certain area to ensure that the intersection results in a non-empty set even if the hole in the cell does not extend through the center of the cell. The operation of morphological dilation with a disk structuring element with radius R_A / 2 is used to create disk markers with radius approximately half of the radius of the cell. Radius R_A is the average radius estimate computed by function `getCenters`.

Binary morphological dilation is a morphological operation that can be used to fill small holes, narrow gulfs in objects or increase object size [7]. Dilation $X \oplus B$ combines two sets using a vector addition and can be defined as:

$$X \oplus B = \left\{ p \in \varepsilon^2 : p = x + b, x \in X \text{ and } b \in B \right\} \tag{1}$$

where B is a structuring element, which is in our case a flat disk-shaped element with the specified radius and with origin in the center of the disk, and X is the object to be dilated.

The dual operator of dilation is morphological erosion which combines two sets using vector subtraction of set elements and can be defined as

$$X \ominus B = \left\{ p \in \mathcal{E}^2 : p + b \in X \text{ for every } b \in B \right\} \quad (2)$$

In contrast with dilation, erosion is used to simplify the structure of an objects and causes objects or their parts with width smaller than the width of the structuring element to disappear.

The second operation that is performed before computing the intersection of the two sets is morphological opening with a disk structuring element of radius R_A / 10 which is used to remove small regions of pixels that were identified as holes and filled by the flood-fill operation. These regions do not represent any real holes in the original image but rather occur as a result of noise or artifacts. They are therefore removed and so excluded from the subsequent intersection operation.

Morphological opening $X \circ B$ is defined as erosion followed by dilation [7]:

$$X \circ B = (X \ominus B) \oplus B \quad (3)$$

where structuring element B is in our case again a flat disk-shaped element with origin in the center of the disk. Opening removes objects with width smaller than the width of the structuring element, but does not decrease the size of other objects, although it affects their shape close to the borders.

The algorithm for filling the holes in the cells centers is implemented in the function `fillHoles.m` and can be summarized as follows:

1. Identify the holes in the input binary image I as background pixels that cannot be reached by filling in the background from the edge of the image and fill these holes by inverting the values of these pixels. The resulting image is denoted as I_f.
2. Compute the difference between the filled image I_f and the original image I to obtain an image of only the filled regions: $I_h = I - I_f$.
3. Open the image I_h with disk-shaped structuring element SE with radius $r = R_A$ / 10 to remove filled holes smaller than 1/10 of the cell: $I_{ho} = I_h \circ SE$
4. Create a marker image I_m of the size of image I_h by setting the value of a pixel to 1 if the location of the pixel is an element of the set M containing the indexes of estimated centers of red blood cells:

   $I_m(i,j) = 1$ if $(i,j) \in M = \{(i_{center1}, j_{center1}), (i_{center2}, j_{center2}), \ldots\}$; else $I_m(i,j) = 0$

5. Label connected regions in I_h and create sets of pixels I_{h_1}, I_{h_2}, ... representing individual holes.
6. For each I_{h_i} compute intersection with I_m: $I_{h_i m} = I_{h_i} \cap I_m$.

   If $I_{h_i m} = \emptyset$, revert values of pixels of a corresponding filled region in I_f from 1 to 0. Such region represents a filled hole in the original image I which is not in the center of any cell.

6.2.2 Identifying Single Cells and Cell Compounds

The aim of the segmentation is to isolate each individual red blood cell in the image. Some connected regions in the binary image of the red blood cells with filled holes already represent individual cells. The rest of the regions are overlapping cells that form clusters which need to be separated. To distinguish between single cells and cell compounds, two simple criteria with empirically estimated parameters were used: the relative area and elongation of the object. In this step, also objects that are too small to represent a cell are removed.

The area of an object is computed simply as a sum of pixels of the object. This value is then normalized by the area of a circle with the average cell radius which was estimated in the previous steps. The area is, however, not computed directly from an object but from its convex hull. Convex hull can be defined as the smallest region which contains the object, such that if we connect any two points of the region with a straight line, all points of the line will belong to the region [7]. Since most single red blood cells in the blood slide image have circular or elliptical shape, their area will be approximately equal to the area of their convex hull. In some cases, however, a cell in the thresholded image appeared as a partially open circle (Fig.5a). This could happen due to the variations in intensities within cells in the image or it could be caused by artifacts. Since the inner area of such a cell, which was segmented by thresholding as background, is not surrounded and enclosed by pixels of foreground, it could not have been identified as a hole and filled in the previous step. By computing the convex hull of the cell, we can not only better approximate its real area (which could be otherwise too small and the object could be removed for not being a cell) but also partially repair the mask of the cell which is to be used, in case it is recognized as a single cell, as the output of the segmentation process.

a) *b)*

Fig. 5: Red blood cell mask with an open center area a); convex hull of the same mask b)

If the region in the image is a cell compound, the relative area of the convex hull will in most cases be greater than the relative area of the region itself, which further helps to distinguish between single cells and compounds.

Due to the variances in cell radii, the relative area alone is not sufficient to distinguish between single cells and cell compounds. For example, the area of two smaller overlapping cells can be in some cases comparable with the area of a single infected red blood cell. They would, however, differ in the elongation, which was used as the second criterion.

Elongation is a measure frequently used to describe the shape of an object and can be defined as the ratio of an object's length to its breadth [13]:

$$E = \frac{length}{breadth} \tag{4}$$

One way of computing the elongation is based on calculating the ratio of the long side to the short side of the object's bounding rectangle. This method is, however, not sufficient, because elongation calculated in this way is dependent on the orientation of the object.

A better way of computing the object's elongation is to calculate it using second-order moments of the object defining its major and minor axis [12,13]. Elongation in this case can be given by

$$E = \frac{\chi_{\max}}{\chi_{\min}} \tag{5}$$

where

$$\chi^2 = \frac{1}{2}(a + c) + \frac{1}{2}(a - c)\cos 2\theta + \frac{1}{2}b\sin 2\theta \tag{6}$$

and

$$a = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(x - \bar{x})^2 I[i,j] \tag{7}$$

$$b = 2\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(x - \bar{x})(y - \bar{y})I[i,j] \tag{8}$$

$$c = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(y - \bar{y})^2 I[i,j] \tag{9}$$

$$\sin 2\theta = \pm\frac{b}{\sqrt{b^2 + (a - c)^2}} \tag{10}$$

$$\cos 2\theta = \pm\frac{a - c}{\sqrt{b^2 + (a - c)^2}} \tag{11}$$

When the expressions for $\sin 2\theta$ and $\cos 2\theta$ are substituted into Eq. 6, the signs determine whether χ^2 is a maximum or minimum. The minimal value of χ^2 represents the axis of orientation. The center of the object $(\bar{x}, \bar{y})$ is calculated as the average pixel x- and y-location from all the pixels constituting the mask of the object.

Following the definition given by Eq. 5, the computation of object's elongation is implemented in the function `elongation.m`.

The method for separating single cells from compounds is implemented in the function `separateCells.m`. The output of the function comprises two binary images, one containing only single cells and the second containing cell compounds. While the compounds still have to be separated in order to obtain masks of individual red blood cells, the image with single cell requires no further processing. The algorithm that separates single cells and compounds can be summarized as follows:

1. Using Matlab's function `bwlabeln`, label connected regions in the input binary image I to separate disconnected regions of foreground and, using these labels, create sets of pixels I_{O_1}, I_{O_2}, ... representing individual objects in the image (single cells, compounds, and artifacts)
2. For each object I_{O_i} :
   2.1 Compute relative area of an object: $A_{r_i} = \frac{\sum_{p \in O_i} I_{O_i}}{\pi \cdot r^2}$, where r is the estimated average cell radius
   2.2 If $A_{r_i} < A_0$, discard object I_{O_i} and continue with step 2.1 for the object $I_{O_{i+1}}$. A_0 is an empirically determined parameter.
   2.3 Compute object's elongation E_i.
   2.4 If $A_{r_i} < A_1$ and $E_i < E_1$, object I_{O_i} is a single cell, otherwise the object is a compound cell. A_1 and E_1 are empirically chosen parameters.

6.2.3 Separating Cell Compounds

To separate individual cells within clusters, watershed segmentation based on distance transformation of the binary image containing cell compounds is used [7,10]. Watershed-based methods are often used for particle segmentation in biological and medical image analysis and different techniques have been proposed and used in previous works [2,3,14].

The term watershed is used in topography and refers to ridges that divide area drained by different river systems. Catchment basins divided by watershed lines are geographical areas draining into a river or reservoir. Watershed segmentation is based on the idea that image data may be interpreted as a topographic surface where the local minima of gray level (altitude) yield catchment basins. Watershed transformation creates an image of regions corresponding to catchment basins of the topographical surface. For segmentation

purposes, gradient images are often used to compute watersheds. Using gradient images, region edges in the original gray-scale image correspond to high watersheds and low-gradient regions correspond to catchment basins. The raw watershed segmentation, however, produces severely over-segmented images. Regions markers and other approaches are usually used to overcome this problem [7]. Watershed transformation can also be used for the segmentation in binary images to find regions corresponding to individual overlapping objects. In such case, the original binary image can be converted into gray-scale using the negative distance transform. The distance function $\mathrm{dist}_X(p)$ associated with each pixel p of the set X is the shortest distance between pixel p and background X^C and using operation of morphological erosion (see section 6.2.1, Eq.2) distance function can be defined as:

$$\forall p \in X \quad \mathrm{dist}_X(p) = \min\{n \in N, p \text{ not in } (X \ominus nB)\} \tag{12}$$

i.e. the distance $\mathrm{dist}_X(p)$ is the size of the first erosion of X that does not contain p. The negative distance transformation –dist has to computed, so that the most distant pixels, which are located in the centers of the cells, represent the basin (i.e. the local minima in the gray-scale image have to correspond to the most distant pixels). The watershed segmentation based on the distance transformation is illustrated in Fig.6.

The standard watershed segmentation in binary images using distance transformation shows good results on cells with circular shapes and smooth contours. However, the shapes of red blood cells vary due to different reasons. In such cases, the watershed transformation may produce over-segmented regions, which is a common problem. In this work, watershed-segmented regions are further analyzed and, based on their areas and shared borders, they are connected back together if a region is identified as too small to represent a cell. Each segmented region is analyzed and if its area is found to be too small to represent a red blood cell, it is connected with a neighboring region with which it shares the longest border. The two regions are, however, connected only if the maximal distance between the contour points of the new object is lesser than an empirically set threshold. This additional condition prevents the regions from being merged in case the resulting object would not represent a red blood cell. Such situation occasionally occurred, for example when a cell was not correctly thresholded in the original gray-scale image.

The number of over-segmented regions can also be, in some cases substantially, lowered by using chessboard distance transform instead of standard Euclidean as concluded in [15], which might simplify the post-processing task. Using this transform, however, some under-segmented regions were also produced containing cell compounds which had not been separated. Since it is using the developed GUI in our case much easier to connect two over-segmented parts of a cell than to manually draw a boundary between two under-segmented cells, the Euclidean distance is preferred. Moreover, the standard Euclidean distance transform produced more natural boundaries between segmented regions and using the post-processing of the segmented regions, it generally produced better results.

The watershed segmentation algorithm with the subsequent post-processing of the segmented regions is implemented in the function `watershedDist.m`. The algorithm can be described as follows:

1. Compute negative distance transformation I_D of the objects in the input binary image I (using Matlab's function `bwdist`):

   $I_D = -\text{dist}_X(I)$

2. Set all background pixels in I_D to *–inf* to impose minima on the background and thus ensure watersheds at objects' edges.

3. Compute watershed transformation (using Matlab's function `watershed`):

   $I_W = \texttt{watershed}(I_D)$

4. For each segmented object $X_i \subset I_W$:

   4.1. Compute relative area of X_i : $A_{r_i} = \frac{\sum_{p \in O_i} X_i}{\pi \cdot r^2}$, where r is the estimated average cell radius

   4.2. If $A_{r_i} > A_1$, continue with step 4.1 for object X_{i+1}

   4.3. Find sets of border pixels B_{N_j} with the neighboring objects N_j

   4.4. Find object $N_{j_{\max}}$ with maximum number of border pixels $\max_j(B_{N_j})$

   4.5. Compute contour of the merged object $C_i = N_{j_{\max}} \cap X_i$

   4.6. Find maximum distance between points of the contour

   $$d_{i\max} = \max_{m,n \in C_i}\left(\sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}\right)$$

   4.7. If $d_{i\max} < d_1$, merge segmented object X_i with the neighboring object N_j

Parameters A_1 and d_1 are empirically set values.

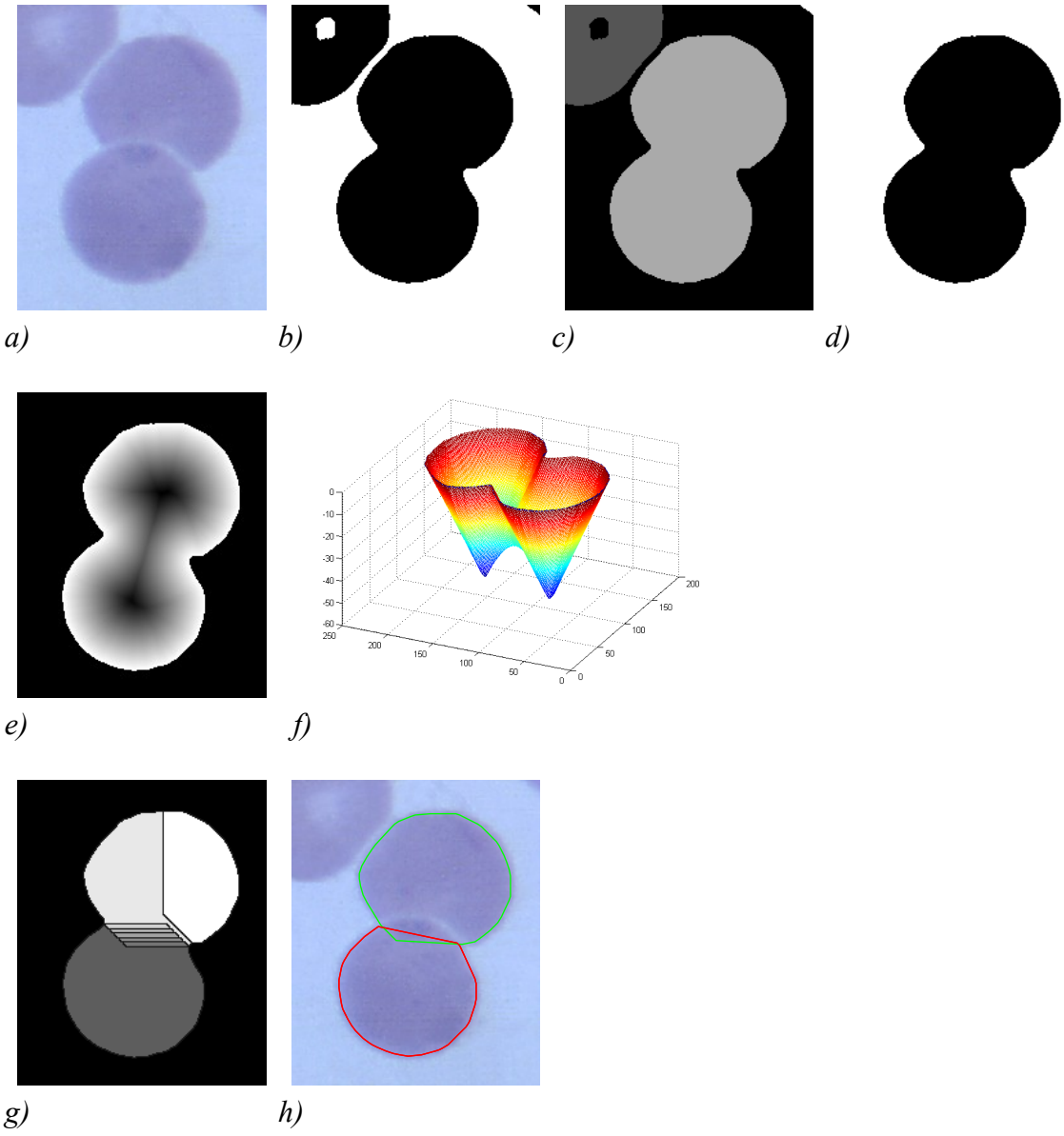


Fig. 6: Separation of overlapping cells using watershed transformation based on distance transformation

a) Original image of overlapped red blood cells
b) Binary image obtained by thresholding image a)
c) Labeling of the connected areas in the binary image
d) Selecting mask of the overlapped cells (based on the area and elongation)
e) Negative distance transform on inverted binary image with imposed minima on background pixels
f) Negative distance transform image (image c) displayed as topographic surface in 3-D, where the third dimension (altitude) corresponds to the gray level values
g) The result of watershed transformation with typical over-segmented parts
h) Resulting contours of separated red blood cells

6.2.4 Contours of Red Blood Cells

From the previous methods, we have two binary images, one with regions representing single cells and the second one containing red blood cells separated by the watershed technique. The contour points, which are one of the outputs of the segmentation function, are not calculated directly from the the binary mask images of segmented red blood cells, but, instead, from their convex hull (see section 6.2.2). Since a red blood cell has typically a circular or elliptical shape, which is a convex shape, the convex hull of most red cells in the image is equal to the original cell shape. This operation, however, helps to correct shapes of cells which were not optimally thresholded and their shape contain gulfs or holes. Moreover, since the watershed segmentation does not produce optimal cuts (Fig.6g), the convex hull helps to better approximate the real shape of the cell (Fig.6.h).

Convex hull is calculated using Matlab's function `convhull`. This function returns indices of the points on the convex hull which are used as contour points of the red blood cells.

6.3 Additional Issues

Two more problems occasionally occurred in certain images when using the segmentation method described in the previous sections. These issues had to be additionally corrected. To ensure the repeatability of the results on the images in which red blood cells had already been segmented using the original method and stored in the database, the original algorithm was not changed, but instead two other variants of the segmentation method were created. Additional processing steps in the original algorithm can be activated by supplying specific input parameters to the implementing function. Another reason for keeping the option to segment the cells using the original method was motivated by the fact that there was a certain possibility that the changes made would not produce optimal results on the standard set of images.

The additional problems occurred mainly because the input images used for the creation of the database were from diverse sources and varied in many characteristics, including variations in illumination, hue, contrast and scale, noise characteristic, smoothness and gradient of the cell margin, and the overall visual appearance of the cells. Although our aim was to propose a universal segmentation method working properly with all input images, providing several variations of a method may also be a good approach. The reason for this is that due to a limited number of input images, the optimality of the method cannot be assessed for all the possible variations of the characteristics of the input image. When providing more variant of the segmentation method, some of them can perform better on a new image than the other. For this reason, the developed GUI provides an easy option of registering new algorithms and selecting between them.

The proposed algorithm did not perform well in two cases – when the contrast in intensity between stained infected red blood cells and non-infected cells was too high and when the image contained many parasites in the early stage of development with well defined edges and red blood cells with rather rough and blurry edges.

The first problem occurred mainly when the image contained one or more red blood cells infected with parasites in later stages of development which were shown in dark saturated color while the rest of the red blood cells were rather pale and shown in light color (Fig.3). When using the Otsu's method for thresholding the gray-scale image, only the pixels of the infected cells were marked as foreground while the rest of the cells were marked as background. Although all the other parts of the segmentation method worked well, only contours of the infected cells were present in the resulting set of segmented red blood cells. To overcome this problem, contrast-limited adaptive histogram equalization (CLAHE) is performed before the gray-scale image is thresholded [16]. The aim of the histogram equalization technique is to create an image with equally distributed brightness levels over the whole brightness scale. Histogram equalization enhances contrast for brightness values close to histogram maxima, and decreases contrast near minima. Standard histogram equalization operates on the whole image. Adaptive histogram equalization, instead, operates on small regions (tiles) in the image and enhances contrast of each tile. The neighboring tiles are combined using bilinear interpolation. CLAHE moreover allows us to limit the contrast enhancement to avoid over-saturation and amplifying of noise in the image especially in homogeneous areas. The number of tiles was set to 8 in both dimensions, uniform distribution was set as desired histogram distribution and contrast enhancement limit was set to 0.05. With this additional pre-processing step, the thresholding method worked properly in all tested images. CLAHE is implemented in function `adapthisteq` which is included in Matlab's Image Processing Toolbox.

The second problem appeared in the function computing the estimated average radius of the objects in the image together with the indexes of the object centers (`getCenters.m`). In images that contained many parasites in the early stage of development, which appeared as small circular dots, and red blood cells with rather uneven cell borders, the edge detector sometimes found relatively small number of edge pixels at the border of the cell and many edge pixels on the contour of the parasite. In some cases, the method in such images returned the radius of the parasites instead of the radius of the red blood cells. Since the parameter of the estimated radius of red blood cells in the image is important for almost all subsequent methods, the wrong value of the radius (which was typically around ten times smaller) caused the following methods to perform inadequately. Typically, the red blood cells in the gray-scale image were suppressed during the succeeding step of illumination correction and, as a consequence, only parasites were present in the thresholded binary image. The output of the segmentation function then typically contained only the contours of the parasites. This problem was corrected by modification of the function `getCenters.m`, which was originally proposed and described in [1]. The function estimates the average radius of red blood cells as a distance which produces maximum number of edge pixels accumulated in certain image locations when the pixels are moved in the direction of their gradient by this distance. A modified version of the method was created that searches for the presence of the second maximum, if the radius found is a small number. If the second maximum complies with certain criteria that are based on its value relative to the value of the first maximum and on the distance between these maxima, the average red blood cell radius is evaluated as the value corresponding to the second maximum. The modified version of the method is implemented in the function `getCenters2.m`.

6.4 Description of the Implementing Function

The segmentation method is implemented in the function `RBCsegmentation.m` and can be summarized as follows:

1. * Convert input image from RGB to gray-scale
2. Filter image with median filter
3. * Calculate locations of cell centers and estimate the average red blood cell radius (using either function `getCenters` or `getCenters2`)
4. Correct non-uniform illumination
5. * Equalize histogram using adaptive histogram equalization
6. Convert image to binary representation using thresholding
7. Fill holes inside the cells
8. Identify and separate single cells and cell compounds
9. Separate overlapping cells using watershed transformation
10. Calculate red blood cell contours and create output data structure

(Steps marked by * depend on additional input parameters or input data format.)

The output of the function is a cell array which contains information about the contour of each segmented red blood cell in the image. When the function is used without any parameters, a dialog box is opened to choose an image file to be processed and a figure displaying the original image with the contours of segmented cells is shown automatically after the segmentation is finished. If the first parameter is a matrix representing the input gray-scale or RGB image, the segmentation of this image is performed. Additional parameters can be provided to specify the file name and path of the image to be loaded, to show images of individual processing steps, or to set the modification of the method to be used, specifically whether to use the adaptive histogram equalization or whether to use the modified function `getCenters2.m`. If neither the input image nor the filename parameter is provided, a dialog box is automatically opened to select an image file. The function checks the inputs using Matlab style. If wrong parameter is supplied, a message with details about the type of expected input parameter is displayed. The details on function usage are given below.

6.4.1 Syntax

```
RBCsegmentation

RBCsegmentation(I)

RBCsegmentation(filename)

RBCsegmentation(I,param1,val1,param2,val2...)

RBCsegmentation(filename,param1,val1,param2,val2...)

DB = RBCsegmentation(...)
```

6.4.2 Description

`RBCsegmentation` segments red blood cells in a blood smear image and computes their contours. When no input parameters are supplied, an open dialog box is automatically shown to select an image file to be processed. If no output parameter is specified, a figure is opened at the end of the segmentation process displaying the original image with the contours of segmented cells.

`RBCsegmentation(I)` segments red blood cells in gray-scale or RGB image `I`. If `I` is a gray-scale image, the initial preprocessing step converting the image from RGB to gray-scale is skipped.

`RBCsegmentation(filename)` segments red blood cells in an image stored in the file specified by the string `filename`. If the file is not in the current directory, or in a directory on the MATLAB path, the full pathname or the pathname relative to the current directory have to be specified. If the string `filename` is an empty string, an open dialog box is automatically shown. This is an alternative to providing no parameters which can be useful when specifying additional parameters described below.

`RBCsegmentation(filename,param1,val1,param2,val2...)`

`RBCsegmentation(I,param1,val1,param2,val2...)`

specifies any of the additional parameter/value pairs listed in the following table. The case of the parameter does not matter. All of the parameter values are logical types so that they can be either `true` or `false`. If the value is a numeric type, it is converted to a logical type so that 0 is considered as `false` and any non-zero value is considered as `true`.

Parameter	Value
`'ShowImages'`	If `true`, images resulting from the individual processing steps are displayed. This option is mainly for educational and debugging purposes. The displayed images include the gradient image, image with edges, illumination correction mask, raw thresholded binary image, watershed transformation, the original image with contours of segmented cells and a few others. Default: `false`
`'UseGetCenters2'`	If `true`, the modified version of the method estimating of the average red blood cell radius (function `getCenters2.m`) is used (see section 6.3) which checks for the presence of the second maximum if the radius found is too small. Default: `false`
`'UseAdaptHistEq'`	If `true`, contrast-limited adaptive histogram equalization is performed before the gray-scale image is converted to binary using thresholding (see section 6.3). Default: `false`

Tab. 1: Description of the parameter-value pairs used as input arguments of the function RBCsegmentation

```
DB = RBCsegmentation(...)
```

returns a cell array in which each item is a structure array containing information about the cell and mask of the segmented cell. The fields of the structure array are specified in Tab.2.

`DB{i}.x1`	x-coordinate specifying the beginning of the window containing the `i`-th segmented red blood cell. This window is computed as the cell's minimum bounding rectangle.
`DB{i}.x2`	x-coordinate of the end of the window
`DB{i}.y1`	y-coordinate of the beginning of the window
`DB{i}.y2`	y-coordinate of the end of the window
`DB{i}.img`	Binary image with the mask of the segmented cell. The position of the mask in the original images is specified by `x1`, `x2`, `y1`, and `y2` and the size of the mask is [`x2` – `x1` + 1, `y2` – `y1` + 1].
`DB{i}.cont`	Two-column matrix with the contour points of the segmented object's mask. Each row of the matrix contains the x- and y-coordinates of a contour point.
`DB{i}.cellRadius`	The value of the estimated average cell radius. This value is stored merely for the computational convenience and is the same for all indexes `i`.

Tab. 2: Description of the fields of the structure array generated for each cell as the output of the segmentation function

6.4.3 Class Support

The input gray-scale or RGB image `I` can be of class `uint8`, `uint16`, `int16`, `single`, or `double`.

6.4.4 Examples

```
RBCsegmentation(I,'showImages',true)
```

Segments input gray-scale or RGB image `I` and displays the images of the results of the intermediate processing steps.

```
RBCsegmentation('','useGetCenters2',1,'useAdapthisteq',1)

RBCsegmentation('','useGetCenters2',true,'useAdapthisteq',true)
```

Since an empty string is provided as the image filename, an open dialog box is automatically shown to select the image file to be segmented. Segmentation is performed using the modified function for estimating the average cell radius and using the adaptive histogram equalization. Only the figure with final results of the segmentation showing the original red blood cell image and the contours of segmented cells is displayed.

```
DB = RBCsegmentation('c:\images\image1.jpg')

DB = RBCsegmentation('..\image1.jpg')
```

Image file specified by an absolute or relative path and a file name is segmented using the default segmentation parameters. The information about segmented cells is stored in the structure array `DB` and no figures are displayed.

Fig. 7: Output of the segmentation method

7 Red Blood Cell Segmentation GUI

7.1 Motivation

In order to be able to evaluate the performance of the proposed features, we need to create a set of red blood cells with sufficient number of both infected and non-infected samples. To create such set of cells, or a database, we need to both segment the cells in the original image and label the segmented objects according to the class to which they belong. It proved not to be a trivial task to develop a segmentation technique that would not require manual correction of the results. This task was even more difficult since we used images with various characteristics, compared to other works where images obtained under controlled conditions were used, and since we had only a limited number of images with similar characteristics available for testing of the segmentation method. Besides, even if such segmentation method existed, we still need a tool for manual labeling of the samples by an expert. We also required a tool that would enable displaying and editing of any previously created data, so that any contour or any label can be changed, cells can be deleted or all the contours can be recalculated using a new segmentation method. Our aim was also to create a tool that could be easily configured and modified for use in further projects and with other segmentation methods, if necessary.

7.2 Requirements

The red blood cell segmentation GUI was develop based on the following requirements:

1. File manipulation:
    - Open dialog box for selection of the image files to be segmented
    - Saving the information about the contours of segmented cells and their labels into an external file
    - Loading of image contours and labels from the external file if the image was already segmented and the data file exists
2. Execution of the segmentation method. There should be a possibility to register a new segmentation technique or change the one currently used.
3. Displaying results of the segmentation either in form of contours in the original image or as a contour/mask for each segmented cell. The same for images with contour data loaded from the external file.
4. Labeling:
    - Manual assigning and change of labels representing the class of the object
    - There should be a possibility to edit the list of labels

5. Editing of contours/masks:
    - Deletion of contours/masks
    - Editing of contours/masks
    - Possibility to manually create a new contour of a cell

7.3 Creating a GUI in Matlab

Creating the GUI in Matlab has several advantages. Mainly, all the other developed Matlab functions can easily be used by the GUI and the GUI itself can be used by other Matlab functions or m-scripts and it can easily be modified so that the program can be reused and changed, if necessary, to be ready to serve in similar tasks in the future. Although it requires adopting some new principles, all the functions of the GUI are implemented using the standard and familiar Matlab m-files.

The drawback of creating a GUI in Matlab is the limited number of components, their properties and their actions. More complicated or otherwise specific problems may thus require rather awkward solutions. Also, it is sometimes not an easy task to find the specifications of certain properties, their values or the actions of the GUI components. Matlab GUI is thus suitable mainly for implementing simple control elements, selection or option boxes, confirming actions with buttons, etc. This is in accordance with our requirements.

It is possible to create the whole GUI with all its components dynamically in a Matlab function or a script. This requires first creating the figure in which the components are to be displayed and then manually creating all the required components using the `uicontrol` function. All component properties, such as the type (style) of the component (i.e. a pushbutton, a check box, axes, etc.), position, or any other parameter values have to be specified within the code. Moreover, for each action, such as a button click or check box selection, a callback function has to be provided and associated with the component. The callback function can also be specified when creating the component using the `uicontrol` function. This approach is suitable mainly for simple GUIs, especially if most of the components have to be, for some reason, created dynamically anyway. Since all the components are created in the m-file, no figure file is needed to store the properties of the components.

For more complicated GUIs, the described approach may be a little tedious, the code might be difficult to read and modified, and the lack of visual control may cause some unexpected results. The easier and usually preferred way of creating a GUI, which was also utilized in this work, is to use the Matlab GUI creator called GUIDE, which can be opened by the `guide` command. GUIDE (GUI Design Environment) is a layout editor that allows one to create or edit GUIs interactively by simply selecting a component and placing it into the figure using the mouse. The editor also includes a property inspector in which we can

easily edit all the properties of the component. By default, the editor also automatically creates a template m-file with generated empty callback functions, which have to be implemented. This file has the same name as is the name of the GUI figure file containing the components definitions. The GUI thus consists of two files now, the m-file and the figure file. We can run the GUI by executing the function implemented in the generated m-file in the same way we call any other Matlab function and we can even supply the function with defined arguments. The GUI will, however, not work properly by simply opening the figure file.

Callbacks for the components are routines that execute in response to user-generated events, such as mouse clicks and key strokes, and they indicate, for example, that a button was pressed, an item in a listbox was selected etc. All callback functions in the generated GUI m-file have the following standard input arguments: `hObject`, `eventdata`, and `handles`. The first argument `hObject` is the handle of the object, e.g. the GUI component, for which the callback was triggered. It can be used to obtain relevant properties of the object. For instance, `get(hObject,'Value')` can be used to retrieve the toggle state of a check box, option box, toggle button, etc. If we want to change a property of the component issuing the callback, we can use the function `set`: `set(hObject,'PropertyName',PropertyValue)`. The second argument `EventData` is a stream of data describing user gestures, such as key presses, scroll wheel movements, and mouse drags. For components that provide event data, this arguments is a structure which varies in composition according to the component that generates it. The last argument `handles` is a structure that contains handles to all the objects in the figure. It can be used to retrieve or set parameters of any object in the figure, but also for storing application data.

When GUIDE generates the template, it creates the callback function with the name consisting of the component's `Tag` property, underscore, and one of the component's callback properties, which is usually '`Callback`'. The Tag property is a unique name of the component which identifies a component within the GUI. For instance, a callback function automatically generated for a pushbutton with the Tag `pushbutton1` has the following syntax:

```
function pushbutton1_Callback(hObject,eventdata,handles)
```

The component may have other callbacks, for example a `CreateFcn` or a `DeleteFcn`, which GUIDE populates in the same way.

The `handles` structure is maintained as `GUI data` which is a mechanism for sharing property values and application data. We can use the function `guidata` to store a single variable associated with the GUI. When using GUIDE, this variable is the `handles` structure which is automatically passed as an input argument to every callback function and, therefore, it is typically used to store any application data that should be accessible to other callback functions. The application data is stored as a value of a field of the `handles` structure. An important thing to do after any changes are made to the `handles`

variable inside the callback function is to update the variable by calling `guidata(hObject, handles)`. This function is typically called at the end of each callback function, because without it all the changes in the `handles` variable are lost. Function `guidata(object_handle,data)` stores the variable `data` as GUI data. If `object_handle` is not a figure handle (in our case it is usually a component handle `hObject`), then the object's parent figure is used.

Matlab GUI can consist of fourteen available components: push button, slider, radio button, check box, static text, edit text, pop-up menu, listbox, toggle button, table, axes, panel, button group, and ActiveX component. Axes enable the GUI to display graphics such as graphs and images. Panels allow GUI components to be visually grouped. Button groups are similar to panels but also allow managing exclusive selection behavior for radio buttons and toggle buttons. ActiveX components enable displaying ActiveX controls in the GUI.

Matlab GUI also enables creating of menus and custom toolbars. We can create menu bars with pull-down menus as well as context menus that can be attached to components. In addition to creating a new toolbar, it is also possible to display (or hide) the Matlab standard figure toolbar, which can be modified to display only desired tools.

7.4 Data Structure Specification

Contours of segmented red blood cells are stored in a variable named `dbx` which is saved in an external MAT-file. Matlab MAT-files are compressed binary files which are used for saving Matlab variables and can be easily loaded by any Matlab function or script using function `load`. One such file is created for each processed image and the file contains information about the contours of all segmented red blood cells in the image. The MAT-file has the same name as is the filename of the image (without the extension) and is saved in the same directory as the image. For example, if the image filename is '2710.tiff', the data will be stored in the file '2710.mat'.

Variable `dbx` is a cell array with the length equal to the number of segmented objects in the image. Each cell contains a structure array with the fields specified in Tab.3.

The structure may contain also other fields created, for example, by the segmentation method (see section 6.4.2), but only the fields specified above are used by the GUI. Any other fields are preserved and saved without any change. Moreover, the field `desc` is not required, but it will be added to the structure array when the corresponding red blood cell is labeled.

`DBX{i}.x1`	x-coordinate specifying the beginning of the window containing the `i`-th segmented red blood cell.
`DBX{i}.x2`	x-coordinate of the end of the window
`DBX{i}.y1`	y-coordinate of the beginning of the window
`DBX{i}.y2`	y-coordinate of the end of the window
`DBX{i}.cont`	Two-column matrix with the contour points of the segmented object's mask. Each row of the matrix contains the x- and y-coordinates of a contour point.
`DBX{i}.desc`	A string with the description (label) of the cell specifying the cell's class.

Tab. 3: Required fields of each structure array in the cell array variable `dbx`.

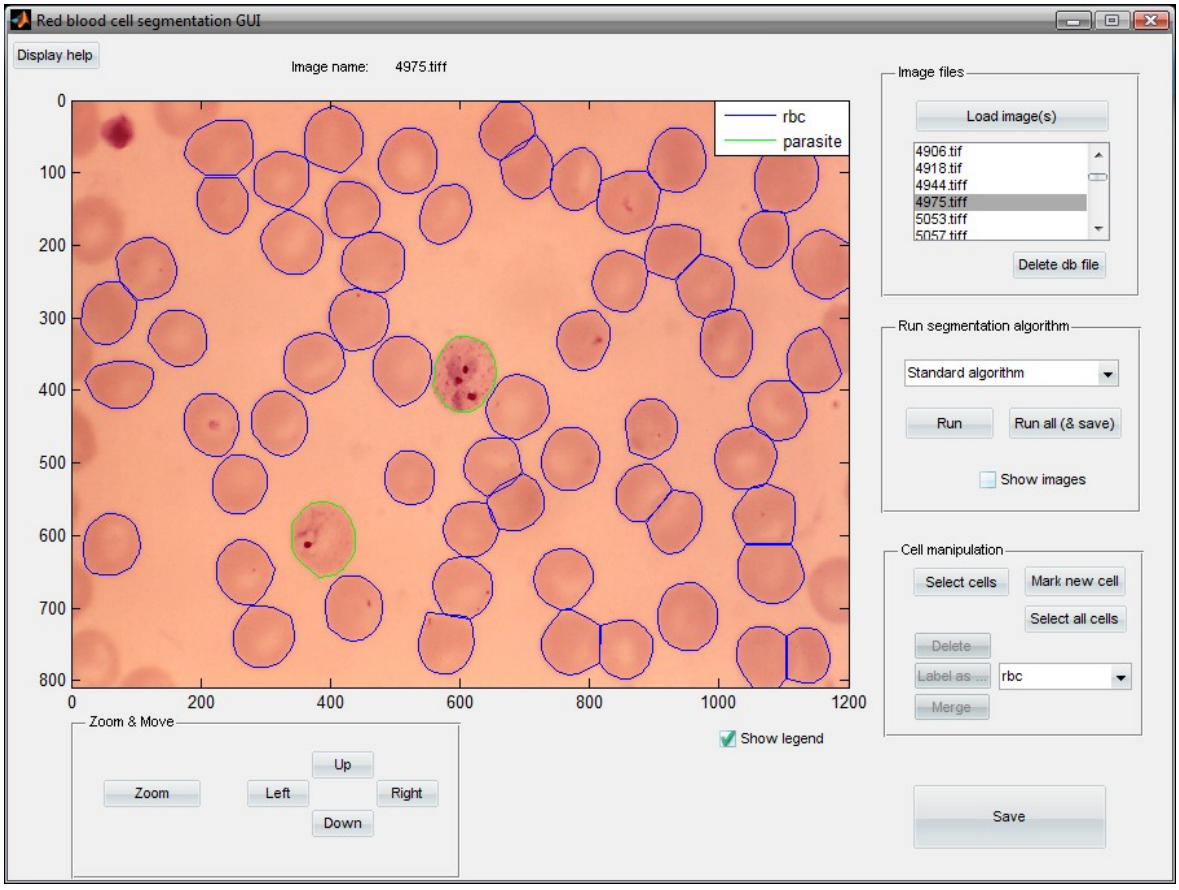


Fig. 8: The red blood cell segmentation GUI

7.5 Program Description

The GUI window (Fig.8) is visually divided into five parts. The main portion of the window consists of the axes component which displays the original image with the contours of segmented red blood cells and is also used for any operation with the cells (deletion, labeling, creating new contour, …). The axes also display the size of the image in pixels, so that we can easily see the resolution of the image and the real size of the zoomed region in pixels. The four remaining panels group components with similar functions. Panel 'Zoom & Move' enables zooming in the image and moving the zoomed window across the image. Panel 'Image files' enables fast switching between loaded image files. In the panel 'Run segmentation', we can select and execute the segmentation method. Panel 'Cell manipulation' groups all the functions for correcting the segmentation results.

After executing the GUI m-file, the figure is automatically opened and the initialization function (named `OpeningFcn`) is called. This function is executed just before the GUI figure is made visible. Besides the standard three input arguments `hObject`, `eventdata`, and `handles`, where `hObject` is the handle to the figure, the opening function also has the argument `varargin`. This cell array is a variable length input argument list which contains optional input arguments passed to the main GUI function. By default, the GUI m-file accepts certain input arguments that allow executing nested callback functions. Additionally, the function also accepts an argument specifying the directory from which the images are to be loaded (see section 7.6 for more details). If the argument with valid directory pathname is provided, the initialization function scans the directory and loads all images of the following file types: .jpg, .png, .tif, .tiff, and .bmp. After the GUI figure is opened, the images found in the directory are displayed in the listbox and the first image is automatically displayed in the axes component. Furthermore, if the corresponding mat-file is found, the contours of the segmented red blood cells are automatically displayed as well. The initialization function also loads the list of segmentation methods and the list of labels which are to be displayed in the corresponding pop-up menus (see section 7.8).

The opening function also sets the `output` field of the `handles` structure to be the handle to the figure, i.e. it assigns to it the value of input argument `hObject`. The `handles.output` variable is used by the output function (`OutputFcn`). This function returns specified output to the command line and is executed when the opening function returns control and before control returns to the command line. Thus, the outputs have to be generated in the opening function, or function `uiwait` has to be called in the opening function to pause its execution while other callbacks generate outputs. The handle to the figure is the only output variable used by this GUI. Having the figure's handle enables us to set and query the values of the figure's properties by using the functions `set` and `get` and to control the behavior of the figure, for example remove it by using the `close` function.

7.5.1 File Manipulation

If no input arguments are supplied, the GUI will open with no images loaded. The images to be segmented can be opened by clicking on button 'Load images', which will open standard dialog box for retrieving files. We can select one or multiple image files of the following types: .jpg, .tif, .tiff, .png, or .bmp. After the image files are selected, their names will be displayed in the listbox located below the button and the first image will be automatically shown in the axes including the contours of segmented red blood cells if the corresponding mat-file is found. The current image can be changed at any time by selecting another file from the listbox. The red blood cell contours are always displayed automatically if the mat-file exists. However, when a new image is selected either from the listbox or using the dialog box, any changes made to the contours of the current image are lost, if they have not been saved.

There is one more component located on the right top panel. It is the button 'Delete db file' which can be used to delete the mat-file with the cell contours of the current image. It has the same effect as if the mat-file with the same name as is the name of the image (excluding the extension) was deleted from the directory where the image is located. The manual deletion of the database files may be, however, more useful when large number of files has to be deleted without the need of visual inspection.

The 'Delete db file' button is used mainly when executing the segmentation method for all files loaded in the listbox (using the 'Run all' button). Since this function processes only image files for which the corresponding mat-file does not exist, the database files can easily be deleted using this button if the current segmentation method did not produce good results and these images can be segmented again using another segmentation method.

7.5.2 Segmentation

The middle panel contains four components for setting and executing the segmentation method. The pop-up menu in the upper part of the panel lists the names of registered segmentation methods. The functions implementing these methods and their names are configured in a separate m-file (see section 7.7 on how to configure this file). The two buttons are used to run the segmentation algorithm. Button 'Run' executes the selected segmentation method only for the current image. During the execution, most of the components are disabled and cannot be used. The red blood cell contour data is not saved automatically when the segmentation method is finished, so that we can safely run the segmentation without rewriting the saved data. If we are not satisfied with the results, we can simply discard them by selecting another image file or closing the GUI. However, if we wish to save the results of the segmentation or any changes made, we have to always press the 'Save' button.

Since the segmentation may be a time consuming process, we usually first wish to run the segmentation for all selected images without any user interaction and then review the results and perform any necessary corrections. Button 'Run for all & save' executes the segmentation method for all image files listed in the listbox and automatically saves the result in the corresponding mat-files. However, only images for which the database mat-

file does not exist are segmented. This is to protect the already created and corrected contours from being rewritten. If we wish to run the segmentation method again for images that already contain the database file, we can either use the 'Run' button to segment only the current image and then rewrite the saved data by pressing the 'Save' button or we can delete the mat-files one by one by the button 'Delete db file' and then run the segmentation method for all these images using the 'Run for all & save' button.. Alternatively, we can delete the corresponding mat-files manually in the image directory.

When the segmentation is run for all image files from the listbox (using the 'Run for all & save' button), a progress bar window is displayed showing the name of the currently processed file and a bar indicating the percentage of files already processed. The window is modal, so that it prevents any other components from being used during the computation. The window also contains a 'Cancel' button. When the user clicks on the 'Cancel' button or the close button of the figure, the segmentation process is aborted. The calculation, however, does not stop immediately. The user has to wait until the segmentation of the current image is finished. Moreover, during the segmentation process textual information is displayed in the Matlab command line window with the number of currently processed image, total number of images to be processed, name of the image file and description of the operation performed. Also, the images in the GUI figure are updated during the process to show the currently processed image.

The last component on this panel is the checkbox 'Show images'. The value of the checkbox is passed as an argument to the segmentation method (see section 7.8 for more details about the requirements on the segmentation function). If the box is checked, the segmentation method is supposed to display figures with results of individual processing steps. The type and the number of images shown is solely dependent on the segmentation function. This feature is intended mainly for educational and debugging purposes. If the segmentation method does not perform well on a particular image, we can toggle this feature on to easily identify which processing step is responsible for the inferior performance. The images can also clearly demonstrate how the segmentation technique works, what are its weaknesses, and which steps could possibly be improved.

7.5.3 Correction and Labeling Functions

The right lower panel groups all available tools for correction of the segmentation method results and for labeling of red blood cells. The 'Select cells' toggle button activates a selection mode. The mouse pointer changes to a cross-hair and certain components are deactivated to show that their functions are not available during the selection mode. In the selection mode, we can select a contour in the image by clicking on the area inside the contour using left mouse button. The selected contour changes its color to red. To deselect a contour we can simply click within its area again and the contour will change its color back. The selection mode is ended by right mouse button click anywhere in the figure. However, if we click using the right mouse button within the contour area, this cell will still be selected / deselected as the last one. This approach accelerates the work slightly, because, for instance, if we wish to select a single cell contour, we can simply click on the cell using the right mouse button. The contour selection made is lost if the selection mode is quit and entered again. The selected cells can be deleted, labeled according to the

selected class or merged.

Button ‘Select all cells’ selects or deselects all red blood cell contours in the image. This is useful mainly for changing the labels of all cells in the image.

Red blood cell contours of unacceptable quality can be, after being selected, deleted using the ‘Delete’ button. ‘Label as…’ button assigns a label chosen in the pop-up menu right next to the button to all selected cells. Each label represents a certain class of objects and is associated with specified color and line style of the contour. After assigning the label to the selected cells, the color and the line style of the corresponding contours will be changed accordingly. The following labels are currently supported: ‘Red blood cell’, ‘Parasite’, and ‘Other’. The list of labels as well as the color and the line style of the associated contours can be configured in separate m-files (see section 7.7 for more details). We can also display a legend by pressing button ‘Show legend’ located at the bottom. This will display a list of all labels together with their line styles inside the axes.

The ‘Merge’ button can be used for connecting over-segmented parts of an object. Similarly as for the deleting and labeling buttons, the function applies only to selected contours. The merged object is obtained by computing the convex hull of the constituting parts, which is based on the presumption that the shape of a cell is convex. This assumption proved to be justified for practically all red blood cells and the computation of the convex hull often corrected minor shape deficiencies which would appear if the merged cell was obtained by simply joining the areas of the parts. Moreover, the convex hull provided a convenient solution for filling the potential gaps between the constituting parts.

The last component on the contour editing panel is the ‘Mark new cell’ button which provides one of the key functions of the whole GUI. If the contour of a cell did not represent sufficiently the real shape of the cell and had to be deleted or if, for any reason, the contour was not generated at all by the segmentation method, the GUI provides the option of manually drawing the contour of any object using the mouse. After entering this mode, the mouse cursor will change to a cross-hair and any left button mouse click within the image will create a contour point. Right mouse click within the image will create the last contour point and exits the drawing mode. Right mouse click inside the GUI figure but outside the image exits the drawing mode without creating the last point. The first and the last created point (as well as each two other neighboring points) are automatically connected by a line to produce a closed contour. We can exit the contour marking mode at any time by pressing the Escape key. In such case the already created part of the contour will be discarded. Although we cannot use the ‘Zoom’ button during the drawing, we can move the zoomed window using the ‘Left’, ‘Right’, ‘Up’, and ‘Down’ buttons available on the ‘Zoom & move’ panel.

7.5.4 Changing the View

The last panel located in the left bottom corner of the GUI figure contains buttons for zooming and moving of the zoomed window in the image. The zoom function was introduced mainly for the purpose of creating more precise contours when manually marking an object’s contour using the ‘Mark new cell’ button. It can be, however, also

useful for more convenient object selection or merely for visual inspection of the image. After entering the zooming mode, we can zoom in by clicking anywhere within the image using the left mouse button. The zoom mode allows zooming in only one step at a time and for repetitive zooming we always have to press the 'Zoom' button again. After zooming in, the image is automatically centered according to the zooming point. Right button mouse click within the image will zoom the image to the original view and any mouse click in the GUI figure outside the image or any key press exits the zoom mode. The four buttons 'Left', 'Right', 'Up', and 'Down' allow us to move the zoomed window in the corresponding directions. The axes show in pixels the real region of the image currently displayed which can be useful for fast orientation in the image.

7.5.5 Other Functions

The GUI contains three more buttons which are not grouped in any panel. The 'Save' button saves any changes made, including deleting, merging, labeling, and newly created cell contours, to the corresponding mat-file. If the mat-file already exists, all the data is rewritten and if not, a new mat-file is created. As mentioned in the previous sections, the data have to be saved before we select another image file from the file listbox or using the 'Load image(s)' button, otherwise all the changes are lost. When manually correcting the results of the segmentation method, it is a good practice to save the intermediate results of our work using the 'Save' button, because some of the operations, most importantly the deletion and merging, cannot be reverted.

The second button 'Show legend' was also already mentioned. For each registered label, it shows a sample of the line type and color of the contour together with the corresponding text label.

The third button 'Display help' displays description of the GUI and of the function syntax in the Matlab command window. Clicking this button is equivalent to typing help RBCsegm in the Matlab command line and is included merely for convenience.

7.6 Description of the Implementing Function

As mentioned in section 7.3, the GUI consists of two files – the figure file with the extension .fig and the standard m-file. The figure file contains the layout of the GUI including all the components with defined initial values of their parameters. The m-file contains callback function for all components in the figure file, opening and output functions and certain initialization procedures.

The segmentation GUI consists of two main files `RBCsegm.m` and `RBCsegm.fig` and several auxiliary m-files implementing various functions utilized by the main GUI function `RBCsegm`. Description of three of these functions, `getAlgorithms`, `getDescriptions`, and `getContourColor`, is given in section 7.7. These functions are, in fact, small configuration files that contain names of the algorithms and their implementing functions, names of the classes used as labels, and colors and styles of the

contours for individual object classes. We can use these files to easily modify certain GUI parameters. A brief description of the rest of the functions utilized by the GUI is given in Appendix B.

If the figure file is opened, a figure with the segmentation GUI layout will be displayed. In case the corresponding m-file is in the Matlab's current directory or on the Matlab path, the callback functions will be found and executed properly. However, the GUI will not work properly, because the opening function is not executed and the GUI is not correctly initialized. Therefore, it is always important to run the GUI as a standard Matlab function either from Matlab command line or from another m-script or function.

For demonstration purposes, Matlab script `runGUI.m`, which is located in the root directory of the red blood cell segmentation project, can be used for convenient execution of the segmentation GUI. This script will set the directory with the GUI functions as the current directory, opens the segmentation GUI and automatically loads all images from the images/highres3 directory. Moreover, the script will also display the help for the GUI in the Matlab command line window.

7.6.1 Syntax

```
RBCsegm

h = RBCsegm

RBCsegm('CALLBACK', hObject, eventData, handles,...)

RBCsegm('ImageSrc',imageSrc)
```

7.6.2 Description

- `RBCsegm`

  creates a new `RBCsegm` segmentation GUI figure or raises the existing singleton. By default, only one instance of the program (singleton) is allowed to run. In case an instance of the GUI is already running, the figure is focused instead of opening a new one.

- RBCsegm('CALLBACK', hObject, eventData, handles, ...)

  calls the local function named CALLBACK in `RBCsegm.m` with the given input arguments.

- `RBCsegm('ImageSrc', imageSrc)`

  creates a new RBCsegm segmentation GUI or raises the existing singleton and loads all the image files of the types .jpg, .png, .tif, .tiff, and .bmp located in the directory specified by the string `imageSrc`. When a new figure is opened, the names of the image files will be automatically shown in the image file listbox and the first image will be displayed in the axes. If an instance of `RBCsegm` is already

running, the list of image files will be rewritten with the list of files found in the specified directory and the first image will be displayed. All changes that were made in the currently displayed image and that were not saved will be lost.

- `h = RBCSegm(...)`

  returns the handle to a new RBCsegm figure or the handle to the existing singleton

7.7 Modification of the Configuration Files

Three m-files located in the same directory 'gui', where all the segmentation GUI implementation files are located, are intended for easy configuration of the GUI. These files are short Matlab functions including definition of several variables and their names are: `getAlgorithms.m`, `getDescriptions.m`, and `getContourColor.m`. By modifying these files, we can change the names of the segmentation methods which are displayed in the GUI's select segmentation method pop-up menu and the names of the m-files implementing these methods. We can also change the list of labels which are not only displayed in the labeling pop-up menu, but also used in the `.desc` field of the structure array to store information about each segmented object. Additionally, we can also modify the contours line styles and colors associated with individual labels.

7.7.1 Function getAlgorithms

Function `getAlgorithms` consists of the following code:

```
function [alg_src, alg_desc] = getAlgorithms()

alg_src = {'RBCsegmentation1','RBCsegmentation2',...
     'RBCsegmentation3'};

alg_desc = {'Standard algorithm',...

     'Standard alg. + 2nd maxima detection',...

     'Standard alg. + adaptive histogram eq.'};
```

The function returns two cell arrays `alg_src` and `alg_desc` which have to be of the same length. Cell array `alg_src` consists of strings specifying the names of the function implementing the individual segmentation methods. By default, the functions' m-files (i.e. in this case: `RBCsegmentation1.m`, `RBCsegmentation2.m`, and `RBCsegmentation3.m`) are located in the directory 'Algorithms'. They can be, however, in any directory on the Matlab path. If appropriate, we can add any path to the Matlab path using function `addpath`. These functions have to follow certain syntax and support input and output arguments specified in section 7.8. The current mechanism of the

segmentation function call does not support passing additional arguments to the segmentation function. In case we want to use the same segmentation function but we want to display in the pop-up menu several versions of the method with predefined set of parameter values, we have to write a separate function for each version of the method displayed in the pop-up menu. As long as these functions support the specified input and output arguments, they can call any other method with specified set of input parameter values.

The segmentation function call is implemented in this way also in our case. Functions `RBCsegmentation1`, `RBCsegmentation2`, and `RBCsegmentation3` are short auxiliary functions which all call the main segmentation function `RBCsegmentation`. The second and the third functions call the main method with additional parameter-value pairs specifying the modification of the method to be used. In our case, these parameters are: `'UseAdaptHistEq'` and `'UseGetCenters2'` (see section 6.4.2 for more details).

The second cell array returned by `getAlgorithms` function (variable `alg_desc`) contains string values specifying the names of the methods to be displayed in the pop-up menu for selection of the segmentation method. These strings are used solely for this purpose and they should be sufficiently descriptive.

7.7.2 Function getDescriptions

Function `getDescriptions` contains the following code:

```
function descriptions = getDescriptions()
descriptions = {'rbc', 'parasite', 'other'};
```

This function returns a cell array `descriptions`, which can be modified in order to change the list of available labels. Variable `descriptions` consists of strings specifying the names of object classes. These strings are displayed in the label selection pop-up menu and they are saved together with other contour description data in the external file for each labeled segmented object in the image. Since these string are stored in the external mat-files as part of the segmented object description, the `getDescriptions` function should not be edited once a new project is started to prevent possible inconsistencies among the files constituting the red blood cell database.

7.7.3 Function getContourColor

The third configuration function `getContourColor` can be used to change the color and the line style of contours belonging to a particular class. In contrast to the previous two functions, this function is intended for customization rather than configuration and the GUI will work properly even if the colors are not specified for all possible labels (in such case, predefined colors or the default color will be used). This function also contains short

implementation part which should not be edited. The configuration part of the function is as follows:

```
function color = getContourColor(desc)
% --- edit contour colors and line styles here ---
contourColors = {'b','g','c'};
colorWithoutDescription = 'k—';

% --- implementation section, do not edit ---
...
...
```

The function has one input argument `desc`, which is a string containing the class description, and one output argument `color`, which is a string specifier of the contour line style and color. Two variables can be modified to specify the line style. Cell array `contourColors` consists of shortcut strings specifying the style and color of contours belonging to the same class. The index of the string specifier as stored in `contourColors` variable corresponds to the class name index in the cell array returned by `getDescriptions` function. String `colorWithoutDescription` contains a line style specifier which is used for an unknown or undefined class name. If a class with the name specified by input argument `desc` is defined in function `getDescriptions`, the corresponding line specifier with the same index will be returned by `getContourColor` function, provided that the specifier with such index is defined in `contourColors`. If the class name exists but the line specifier with the corresponding index is not defined in `contourColors`, the function will return an empty string. In case the class name specified by `desc` is not found, the function will return the line specifier defined in the variable `colorWithoutDescription`.

The line style and the color specified by `colorWithoutDescription` is also used for contours of objects that have not been labeled yet. In our case, all red blood cells after being segmented by the segmentation function are displayed in the color and line style specified by `colorWithoutDescription`, because the segmentation method does not perform any classification of the objects and the output argument of the function does not contain the `.desc` field with the class description.

The line style and color specifier is a character string which consists of an element specifying the color and an element specifying the line style and its syntax follows Matlab convention used by `LineSpec` function. Combination of the characters listed in Tab.4 are allowed. If the line style character is missing, the default solid line is used.

Character	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

a)

Character	Line Style
-	Solid line
--	Dashed line
:	Dotted line
-.	Dash-dot line

b)

Tab. 4: String specifiers for color a), and line style b), used by function getContourColor

7.8 Requirements on the Segmentation Function

In order for the segmentation GUI to work properly with newly registered segmentation functions, all Matlab functions specified in the `getAlgorithms.m` file have to comply with the following syntax:

```
DB = RBCsegmentation(filename, 'ShowImages', showImages)
```

where `filename` is a string specifying the name and full path of the image file to be segmented and `showImages` is a boolean value indicating whether the algorithm should display results of the intermediate operations. Since this parameter is intended only for setting the visual output of the segmentation function and is not supposed to affect the results of the segmentation in any way, it can be ignored by the function as long as the function does not terminate with error if this parameter-value pair is supplied.

Output of the function must be a cell array of structure arrays, where each structure represents a segmented object. The following fields are required to be present in each structure: `x1`, `x2`, `y1`, `y2`, and `cont`. The first four are scalars and the last one is a matrix with two columns (see section 7.4 for more details). Field `desc`, which is a string containing the name of the object's class, is not required but if present, it can be read by the GUI. In case the name is found in the list of names returned by the function `getDescriptions` the contour color and line style will be set accordingly. If the name is not registered in the `getDescriptions` function, the contour will be displayed with

the style set for an unknown or unrecognized class. (see descriptions of `getContourColor` and `getDescription` function in the previous section).

The structure may contain also any other fields specifying other parameters of the segmented object. These fields will be preserved in the structure and if the object is not deleted and the results are saved, it will be also saved. However, structures of newly created objects with manually drawn contours in the GUI or of the merged objects will contain only the fields specified above.

7.9 Typical Use of the Program

The following procedure proved to be effective during our work. At first, all the images are automatically segmented using the 'Run for all & save' function. In case the segmentation method fails for any images, we can try another segmentation method either by running the segmentation for each file again one by one using the 'Run' button or by deleting the corresponding database files and then running the segmentation again for all these images using the 'Run for all & save' button. In the following step, any inadequately segmented cells are deleted and if there are any over-segmented cells they are merged by first selecting them using the 'Select cells' function and by applying requested operation. All cells that had to be deleted or cells not recognized by the segmentation method at all have to be manually marked using the 'Mark new cell' button provided that they are to be included in the database. The last step involves selecting all cells using the corresponding function and labeling all cells according to the major class of objects in the image. This is followed by manually selecting objects of a minor class and assigning a label to them. Finally, all changes have to be saved using the 'Save' button. It is, however, advised to save also the intermediate corrections to keep the possibility of returning to the previously saved version by reloading the file from the image file listbox.

8 Red Blood Cell Database

Using the designed GUI and following the procedure described in section 7.9, we have created a database of red blood cells containing both non-infected cells and cells infected by malaria parasites.

8.1 Structure and Properties

The database is not implemented using any of the conventional specialized data structures. Instead, it simply consists of a set of original blood smear image files accompanied by corresponding mat-files. These mat-files contain information about all segmented objects in the image including their labels identifying the class of the object. This information is stored in a variable called `dbx`, which is saved in the file. Variable `dbx` is a cell array with the length equal to the number of segmented objects in the image, where each cell contains a structure array with the fields specified in section 7.4. The mat-files have the same name (except the extension) as the original image files and all the image files and mat-files comprising a particular dataset are typically located in a designated directory.

Matlab MAT-files are compressed binary files which are used for saving Matlab variables and can be easily loaded and manipulated by not only any Matlab function or script but also by other programs external to Matlab.

The utilized approach has several advantages:

- *Simplicity*

  Red blood cell samples can be easily retrieved even by an unexperienced Matlab user. Even without Matlab, the cells can be viewed without any specialized software as the original image files are integral part of the database

- *No information loss*

  The image samples are not rescaled, re-compressed, or modified in any other way from the original version. All information contained in the original image is preserved.

- *Efficiency*

  The individual image samples can often be interpreted only in relation to the original blood smear image. Therefore, we required that the original image is available as part of the database. In this context, the presented approach is the most efficient way of saving both the original image and the individual red blood cell images.

- *Extendability*

  A set of fields specified in section 7.4 is required in each cell of the array `dbx` containing the contour data in order for the segmentation GUI and other scripts to

work properly. However, any number of other fields can be added to each structure array containing any user-defined additional information about a particular red blood cell. All these additional fields are preserved without any change when the database is edited by the GUI. Moreover, we can append any other variables to be saved in the mat-files. These could be the whole transformed original images for computational convenience in further processing or any other data. However, these additional variables are currently not preserved when any changes to the contour data are made by the GUI.

- *Flexibility*

  As the data with the contours of segmented red blood cells are stored separately from the actual image data, we can easily create a database containing any transformed versions of the original images as long as the image dimensions are not changed. For instance, we can create new images with corrected non-uniform illumination and normalized colors and save them under the same names to a new folder together with the unmodified mat-files. In the same way, we can create a set of images consisting, for example, of gradient-transformed images, Fourier-transformed images, Gaussian or median filtered image, etc., if such images are desired. All the transformed versions of the red blood cell samples can be retrieved by the same scripts without any change needed and they can even be displayed by the segmentation GUI with the possibility to edit the contour data, although, naturally, the segmentation algorithm may fail on such images.

8.2 Database Content

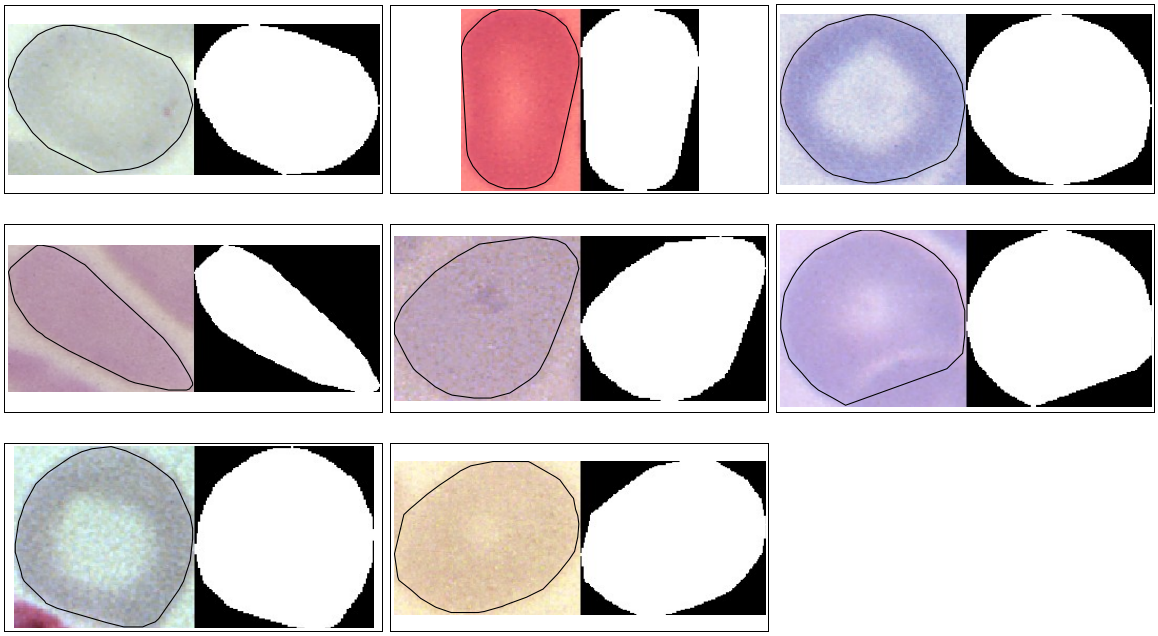
In total, the database contains

- 1811 red blood cell samples with
  - 1694 non-infected red blood cells
  - 117 cells infected by malaria parasites

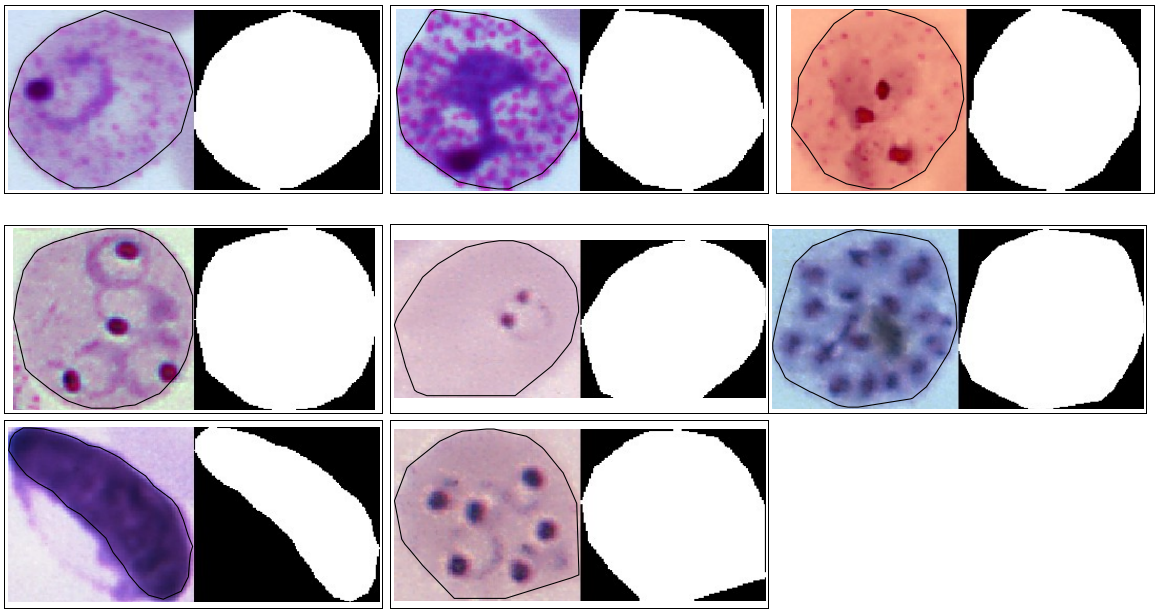
The second group contains mainly red blood cells infected by immature ring-form trophozoites and mature trophozoites of Plasmodium vivax, Plasmodium falciparum, Plasmodium ovale, and, in lesser extent, also Plasmodium malariae. The database also contains samples of P. vivax, P. ovale, and P. falciparum micro- and macrogametocytes and P. vivax schizonts.

The descriptions of the individual blood smear images are included in the project's archive in the directory 'rbcSegm\images\descriptions' (see Appendix B).

Fig.9 shows randomly selected set of both infected and non-infected red blood cell samples generated from the data stored in the database. The mask of the cell is calculated from the contour data.



a)



b)

Fig. 9: Non-infected a), and infected b), red blood cell samples from the created database

9 Feature Extraction

In order to distinguish between infected and non-infected red blood cells, we need to extract features from the image array and compute new variables that concentrate information to separate classes. Such feature set has to consist of features leading to large between-class distance and small within-class variance in the feature vector space, i.e. the set of features should discriminate between different classes as well as possible. An additional requirement is robustness, so that the results can be reproduced for new independently collected material.

Raw images cannot be used directly as features due to high variations in morphology which are coupled with arbitrary rotations and scales and because the raw images contain large amount of data, but relatively little information. This is the aim of feature extraction to transform the input data into a reduced set of features that extract the relevant information from the input data.

Following the concept introduced in [5], the feature extraction process can be expressed in terms of the definition of the *zone of measurement*, an *image transformation* with a non-scalar result and a *measurement* on the latter. This is the process generally followed in this work. Since we were working with the original images, the pre-processing step was added to correct some deficiencies in the input images. Namely, illumination correction, color normalization and noise filtering were performed depending on the particular set of features. The zone of measurement was in our case the whole area of the cell, which was defined by the mask obtained as a result of the preceding segmentation, although in some cases, this mask was eroded in order to remove the border effects in the transformed image. The transformation is used to bring out the aspect of the distribution of the pixel values that is of interest and reflects the aim of the feature extraction method. The transformations used in this work include a histogram, gradient, Laplacian, median filter, co-occurrence matrix and run-length matrix. The transformation results in a non-scalar value, which is usually a new image, a matrix or a vector. The transformations may also be combined so that, for example, a histogram of some measurement from a transformed image can be created and used for the feature calculation.

The final measurement on the transformed image delivers the feature value, which is a scalar. Measurements may be a count, integration, or a selection and since the transformation in many cases results in a histogram, the distribution expressed by this histogram can be characterized by a moment. Similarly, 2D moments may also be calculated to express the shape of the object or the overall density distribution. Through normalization, the measurements can be made robust against several irrelevant variations, such as position, scale, or rotation [19], [20]. It is often the case, that on one transformation, several measurements can be performed and the same type of measurement can be carried out for different transformations. Some features that take only the shape of the cell into account are calculated from the mask of the object. The binary image may as well be seen as a result of a simple transformation, although, in our case, it is the result of the preceding segmentation step.

In some cases, the transformation is not computed using the original pre-processed image, but using an image with applied intensity conversion. For example, we sometimes use the extinction image [5] instead of the original transmission one. Color images usually have to be transformed to gray-scale according to a certain color system, e.g. RGB or HSV. The actual choice depends on the physical properties of the stain used. We used the green channel of the RGB image in the segmentation method, because the Giemsa staining solution has a dark purple color. However, also other types of color transformations are used in this work and the features extracted are evaluated and compared.

In our case, the task is to distinguish whether or not a red blood cell is infected by malaria and, therefore, the selected features must provide information with which it is possible to carry out such classification. When extracting features for the subsequent classification, it is advantageous to apply expert, a priori knowledge to a classification problem [4]. Measures of parasites and infected red blood cell morphology that are commonly used by technicians for manual microscopic diagnosis can be utilized. It is desirable to focus on these features, because it is already known that they are able to differentiate between infected and not infected red blood cells and between species of malaria. Such features, which were suggested in [4], may include, for example, the relative size of the infected red blood cells; the relative eccentricity of the infected cells; smoothness of the cell margin; the relative color of infected red blood cells; and texture information of infected cells, i.e. presence of stippling.

Another set of features can be based on image characteristics that have been used previously in biological cell classifiers [4,5,21]. This set includes various features that are for the purpose of this work grouped into shape features, intensity features, and texture features. The features of the last group have been found the most useful ones in many applications, but at the same time the most difficult to define and the most difficult to understand intuitively [5]. In contrast to the texture features used by technicians for manual microscopic diagnosis, texture features used in image processing and classification are often difficult or even impossible to comprehend and visualize. Another problem with textural features is the missing relation of specific feature values to appearance and function of cells.

The feature set was created by combining different transformations and different measurements on these transformed images with respect to a priori knowledge, observable differences between infected and non-infected red blood cells and achievements in similar studies using the specified feature.

Individual sets of features are evaluated on a created dataset of red blood cell samples using ROC curves for different parameters controlling the feature extraction. Evaluation is followed by a discussion on the effects of different preprocessing techniques and possible utilization of these features for more specific problems of distinguishing between different types of malaria parasites.

The pre-processing of the images is described in section 9.1 and the details on the utilized intensity conversions are given in section 9.2. Definitions and implementation details on the selected features are given in section 9.3 and the feature selection and evaluation process is described in section 9.4.

9.1 Preprocessing of the Images

Since we are working with the original unprocessed images, certain preprocessing steps are necessary to remove differences among individual segmented objects and thus to reduce the variance in the extracted features and improve classification performance. Due to the character of the corrections, these preprocessing steps are performed on the whole input blood slide image before the individual cell images are extracted from it using the information stored in the database.

Although we could as well save the images of individual segmented red blood cells with already applied preprocessing, this approach has several advantages. Firstly, it enables us to always have the original image with the unchanged quality at hand. Secondly, we can perform different preprocessing steps and evaluate their effect on feature's performance. Thirdly, some transformations require no preprocessing because they are invariant to the absolute intensity or color. Two methods are applied in the preprocessing stage. The first is to correct the non-uniform illumination and so to remove the variations in brightness among the red blood cells from a single blood slide image. This method was implemented using the same function which was utilized in the segmentation preprocessing stage and which was described in [1]. The function was, however, modified to work with RGB images instead of gray-scale ones.

The second method applied in the preprocessing stage was color normalization. This is an essential step in order to decrease the effect of different light sources or sensor characteristics and it is especially important if we are extracting any features based on the color of the cell pixels. An adapted gray world normalization method based on the diagonal model of illumination change described in [21,22] was used. Gray world normalization assumes that there is a constant gray value of the image which does not change among different conditions and the diagonal model assumes that an image of unknown illumination $\mathbf{I}^u$ can be simply transformed to known illuminant space $\mathbf{I}^{\tilde{k}}$ by multiplying pixel values with a diagonal matrix: $\mathbf{I}^{\tilde{k}}_{rgb}(x) = \mathbf{M}\mathbf{I}^{u}_{rgb}(x)$. Based on the gray world assumption, if there is an image with known illuminant $\mathbf{I}^k$, the matrix $\mathbf{M}$ can be calculated as follows:

$$\mathbf{M} = \begin{pmatrix} m_{11} & & \\ & m_{22} & \\ & & m_{33} \end{pmatrix} \qquad m_{11} = \frac{\mu^{\mathbf{I}^k_r}}{\mu^{\mathbf{I}^u_r}} \quad m_{22} = \frac{\mu^{\mathbf{I}^k_g}}{\mu^{\mathbf{I}^u_g}} \quad m_{33} = \frac{\mu^{\mathbf{I}^k_b}}{\mu^{\mathbf{I}^u_b}} \tag{13}$$

where $\mu^{\mathbf{I}_{r,g,b}}$ are the means for channels *r*, *g*, and *b*.

Since in our case the background pixels are not used for extraction of features, we normalize the color in the whole image using the means $\mu^{\mathbf{I}^f_{r,g,b}}$ of the foreground computed from the segmented red blood cells.

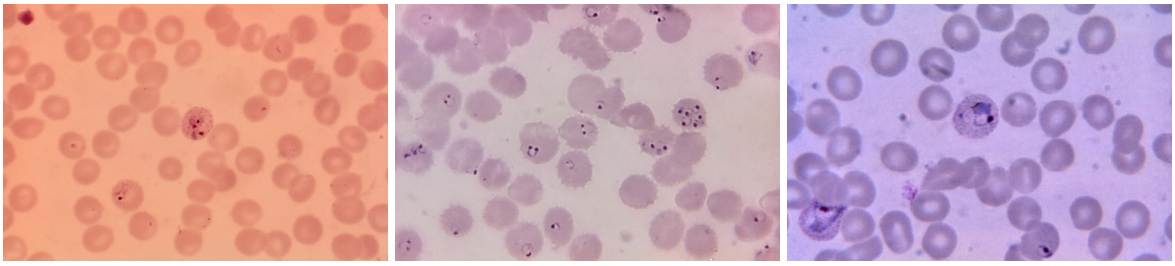
Since red blood cells in many images have ring-like shapes with center intensity and color close to the intensity and color of the background, we do not use the masks of already

segmented red blood cells saved in the database, but instead perform a new segmentation of the image by thresholding the image with automatically estimated threshold to compute the mean values of the *r*, *g*, and *b* channels of the red blood cells unbiased by the background pixel values. The mean values $\mu^{\mathbf{I}^f_{r,g,b}}$ are computed as the means of the pixels marked by the segmentation process as foreground for the individual channels *r*, *g*, and *b*.

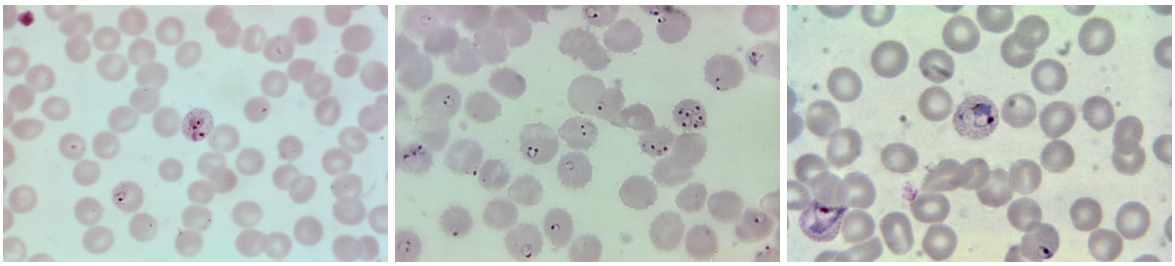
The thresholding of the image using the Otsu's method is preceded by the conversion of the RGB image to gray-scale representation by eliminating the hue and saturation information while retaining the luminance and by applying the adaptive histogram equalization to enhance the contrast in the image, which is useful especially when the contrast between non-infected red blood cells and background is low while the contrast between infected and non-infected red blood cells is relatively high (see section 6.3).

The color normalization method is implemented in the function `colorNormalization2.m` and can be summarized as follows:

1. Convert the original truecolor RGB image I to gray-scale image I_G
2. Perform the adaptive histogram equalization on the gray-scale image I_G
3. Convert the gray-scale image I_G to binary image I_B by thresholding with automatically selected threshold
4. Calculate the mean values $\mu^{\mathbf{I}^f_{r,g,b}}$ of the foreground for channels *r*, *g*, and *b*. Pixels of foreground are identified as pixels in the original image I for which $I_B = 0$: $I^f_{rgb} = \{p \in I : I_B(p) = 0\}$. Calculate $\mathbf{M}^\mathbf{f}$: $m^f_{11,22,33} = \frac{\mu^{\mathbf{I}^k_{r,g,b}}}{\mu^{\mathbf{I}^f_{r,g,b}}}$, where $\mu^{\mathbf{I}^k_{r,g,b}}$ are mean values of the reference image.
5. Transform the whole image I to normalize the color: $I_2 = \mathbf{M}^\mathbf{f} I$



a)



b)

Fig. 10: Color normalization. a) Original images, b) images with normalized colors

9.2 Intensity Conversions

For most of the feature extraction methods, only one intensity value per pixel is assumed, i.e. the methods assume a gray-scale image. For some methods, the original RGB image is converted to gray-scale by eliminating the hue and saturation information while retaining the luminance. For certain features, we evaluated the performance for several channels to choose the channel with best discriminating power. In addition to the red, green, and blue channels, we also evaluated the performance of the feature for saturation, and value channels after transforming the image to the HSV color system. The HSV color system describes colors as points in a cylinder whose central axis ranges from black at the bottom to white at the top with neutral colors between them. The distance along the axis corresponds to *value*, or brightness. The angle corresponds to *hue*, the perceived color, and the distance from the axis corresponds to *saturation*.

Intensity values of the original image represent the transmitted light. However, for calculating some features, the pixels values from the extinction image can be more directly useful [5]. Therefore, a transformed extinction image is created using the following formula:

$$I_E = c_1 \log_{10}\left(\frac{I_T - s}{c_2}\right) \tag{14}$$

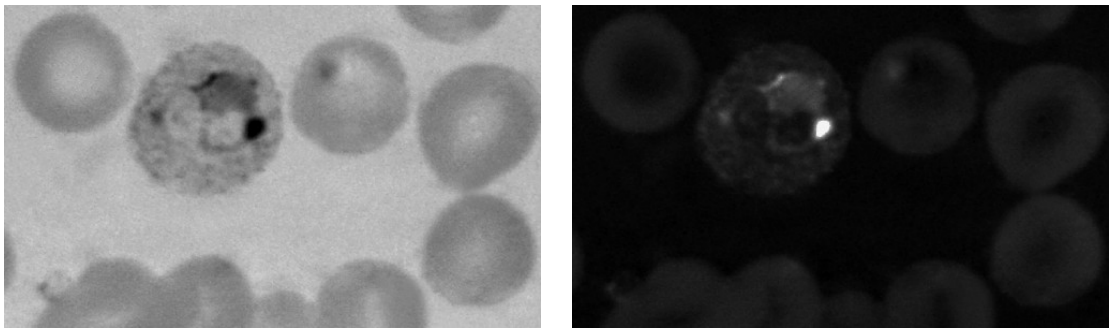
where I_T is the original transmission image, c_1 is a multiplicative scale factor, c_2 is a

preliminary white value, and s is the base value of transmission zero (the black shoulder). In our case, the following parameters were used:

$c_1 = -0.59$

$s = -0.02$

$c_2 = (\max(I_T) - s) / (1 - s)$



a) *b)*

Fig. 11: Green channel of the original transmission image a) and corresponding extinction image b)

9.3 Feature Generation

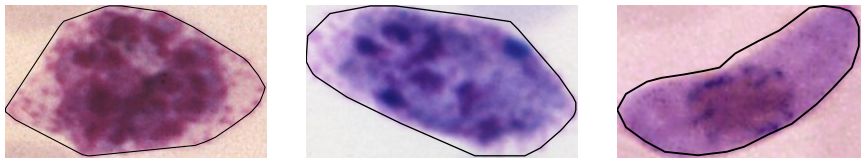
A set of features is proposed and implemented in this stage of the project. The selection of the features for the further evaluation was based on the visual differences between infected and non-infected red blood cells, the measures of infected red blood cells that are commonly used by technicians for manual microscopic diagnosis, and the feature selection used by other cytological studies. The chosen features can be grouped into three categories: shape features, intensity features, and texture features.

9.3.1 Shape Features

These features express the overall size and shape of the cell without taking the density of the cell into account, except for the initial segmentation step. In other words, these features are computed only from the mask of the object which was obtained by the preceding segmentation process and the actual gray-scale image is not needed. Although the applicability of the features based only on the shape of the cell is necessarily limited, they can be useful in distinguishing development stages of certain species of plasmodium which are characterized by a specific shape of the infected cell and they may also be useful in

distinguishing between red blood cells and other objects, such as white blood cells, platelets of artifacts. Although the number of classes had to be reduced to only two classes – an infected and a non-infected cell – due to the insufficient number of samples, several of these features were evaluated both as guidance for possible future studies, which would include also classification distinguishing between infected and non-infected red blood cells and other objects and classification of parasites according to the development stage and species of the plasmodium, and also to evaluate whether these features could possibly improve the overall discrimination power when combined with other features.

The use of shape features was motivated by several observable differences in shape between infected and non-infected red blood cells. One of the most frequent differences was the size of the cells infected by plasmodium in later stages of development which was usually greater relative to the average size of non-infected cells (i.e. also relative to the average cell size in the blood slide due to the majority of non-infected cells in the image). The shapes of cells infected by plasmodium parasite in later stages of development were also often anomalous compared to the relatively regular circular or elliptical shape of non-infected red cells. The most common deviations include elongation of the infected cell or protrusions as a result of the cell rupture.



a) *b)* *c)*

Fig. 12: Infected red blood cells with distinct shapes. a) Mature P. ovale trophozoite, b) immature P. ovale schizont, c) P. falciparum macrogametocyte

A crescent or sausage-like shape is typical for mature Plasmodium falciparum. The red blood cell hosting the parasite is often distorted or not visible and there is often no distinctive texture, although the parasite is shown in dark saturated color. The shape measurements may, in such cases, become useful supplementary features.

The margin of red blood cells infected by plasmodium parasite in later stages of development is also often crenelated. However, the crenelation is difficult to detect from the mask of the object due to the properties of the segmentation method. Generally, any distinguishing properties of the infected red blood cell contours are hardly detectable from the cell mask image due to the limitations in the segmentation method which, in our case, smooths the contour in order to repair incidental shape deficiencies caused by intensity variations, noise, and artifacts. However, the contour crenelation is in most infected cells negligible and thus this deficiency of the segmentation method is acceptable.

One of the problems associated with shape features is that the shape of a segmented red blood cell is partially dependent on the segmentation method used. In our case, the

compound cells are separated using watershed transformation based on distance transformation in the binary image, which simply cuts the cluster in the narrowest part. The resulting cell shapes are not circular and in case the cells are largely overlapped they may become rather half-circle shaped. However, other segmentation methods, or even manual segmentation, could separate cell compounds in different way and, therefore, the shape features should, at least partially, be seen in relation to the utilized segmentation method.

Since absolute measurements, such as orientation, absolute coordinates, and absolute dimension are of no use for us, we have to choose features which are invariant under transaction, changes in scale, and also rotation. Two sets of features which are in accordance with these requirements have been evaluated: Hu set of invariant moment features and a relative shape measurements vector.

9.3.1.1 Hu Set of Invariant Moment Features

Geometric moments describing the extent of the object are useful shape descriptors which are straightforward to define, but unfortunately sensitive to overall size and orientation of the object. Invariant moments which are normalized against these factors were proposed by Hu [19]. These moments are derived from algebraic combinations of the first three orders of normalized central moments and are translation, scale, and rotation invariant while providing spatial information. The moments are defined as follows:

$$I_1 = \eta_{20} + \eta_{02} \tag{15}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \tag{16}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{17}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{18}$$

$$\begin{aligned} I_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \end{aligned} \tag{19}$$

$$I_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{20}$$

$$\begin{aligned} I_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \end{aligned} \tag{21}$$

Moment I_7 is skew invariant, which is useful in distinguishing mirror images of otherwise identical images.

η_{ij} is normalized $(ij)^{\text{th}}$ central moment where $i + j \geq 2$. These moments are invariant to both translation and changes in scale which is accomplished by dividing the corresponding

central moment by the properly scaled $(00)^{th}$ central moment, using the following formula:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(\frac{i+j}{2}+1\right)}} \tag{22}$$

Central moments for a digital image $I(x,y)$ are defined as follows:

$$\mu_{ij} = \sum_{x} \sum_{y} (x - \bar{x})^i (y - \bar{y})^j I(x,y) \tag{23}$$

Since in our case the image $I(x,y)$ is the object's mask, the formula can be rewritten as:

$$\mu_{ij}(\mathbf{1}_O) = \sum_{(x,y)=p \in O} (x - \bar{x})^i (y - \bar{y})^j \tag{24}$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}} \tag{25}$$

are x- and y-coordinates of centroid and m_{ij} are general 2-d moments defined as:

$$m_{ij} = \sum_{x} \sum_{y} x^i y^j I(x,y) \tag{26}$$

Using Eq.26, $\bar{x}$ and $\bar{y}$ can be rewritten for a binary mask image in form:

$$\bar{x} = \frac{1}{A} \sum_{(x,y)=p \in O} x \,, \; \bar{y} = \frac{1}{A} \sum_{(x,y)=p \in O} y \tag{27}$$

Where A is the area of the object. Hu invariant moment features $I_1 - I_7$ are calculated by the function `fHuMoments.m` according to Eq.15-27.

9.3.1.2 Relative Shape Measurements

Relative shape measurements **R** is a vector containing five simple shape measurements which are normalized using the estimated average cell area computed during the segmentation and stored for convenience in the image database for each image file (see section 6.1). These measurements are independent of position and orientation and after normalization they are also approximately independent of size. Let r denotes the estimated average cell radius and O is the object's mask. Then the relative shape measurements used and evaluated as features in this work are calculated as follows:

Relative area

$$A_r = \frac{\sum_{(x,y)=p \in O} 1}{\pi \cdot r^2} \tag{28}$$

Relative area is computed as sum of the mask pixels (e.g. the area of the mask) divided by the area of a circle with radius *r*.

Relative perimeter

$$P_r = \frac{\sum_{i=0}^{N-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} + \sqrt{(x_{i_{\max}} - x_0)^2 + (y_{i_{\max}} - y_0)^2}}{2\pi \cdot r} \tag{29}$$

The relative perimeter is computed as the length of the object's contour divided by perimeter of a circle with radius *r*. Since in our case the contour is not a connected line but consists only of necessary number of points to reconstruct the mask, its length is calculated as sum of distances between adjacent point plus distance between the first and the last point as the contour is a closed line. *N* is the number of contour points.

Major axis of best-fit ellipse

$$a_r = \frac{a}{r} \tag{30}$$

Minor axis of best-fit ellipse

$$b_r = \frac{b}{r} \tag{31}$$

The best-fit ellipse is computed by solving nonlinear least-squares curve fitting problem. However, finding a best-fit ellipse with arbitrary position and rotation is an optimization problem with many parameters and there is a high probability that the local solution found would not represent the desired best-fit ellipse. Moreover, we would have variables in the denominator and numerator which can shrink and grow together and we could obtain very large or very small coefficients. Therefore, we first find the orientation of the object and rotate the object so that its axis of orientation is parallel to the x- or y-axis in the Cartesian plane. Furthermore, we estimate the center of the ellipse $(\bar{x}, \bar{y})$ as x- and y-coordinates of centroid according to the Eq.27. Then we can simplify the optimization problem of finding the best-fit ellipse to the problem of finding the coefficients *a* and *b* in the equation of ellipse:

$$\frac{(x - \bar{x})^2}{a_1^2} + \frac{(y - \bar{y})^2}{a_2^2} - 1 = 0 \tag{32}$$

The larger of the coefficients is the semimajor axis of the best-fit ellipse, the second is the semiminor axis. Thus the major axis $a = 2\max\{a_1, a_2\}$ and the minor axis $b = 2\min\{a_1, a_2\}$.

The orientation is calculated using the following formula [12].

$$\theta = \frac{1}{2}\arctan\left(\frac{b}{a - c}\right) \tag{33}$$

where *a*, *b*, and *c* are the second-order moments (Eq.7-9, section 6.2.2). After computing the object's orientation, the object contour points are converted from Cartesian to polar coordinate system, where position of each point is specified by θ, which is the angular displacement from the positive x-axis, and ρ, which is the distance from the origin to the point in *x*-*y* plane. The calculated angle of orientation is then subtracted from each point and after that all the points are converted back to Cartesian coordinate system.

Largest inscribable circle

$$r_{lic} = \max(dist(I)) \cdot \frac{1}{r} \tag{34}$$

Although all these features are closely related to the size of the object relative to the estimated average red blood cell radius, they all describe different aspects of the shape. While the area is roughly proportional to the overall size of the object regardless of its shape, the perimeter is greater for concave objects, objects with irregular shape or objects with crenelated or otherwise non-smooth contour. The major axis of the best-fit ellipse roughly corresponds to the largest distance within the object. For convex shapes, the minor axis of the best-fit ellipse is similar to the double of the radius of the largest inscribable circle and both express the size of the narrowest part of the object. These values will, however, differ for object with irregular or concave shapes, for example for crescent-shaped cells.

9.3.2 Intensity Features

Intensity features are based only on the absolute value of the intensity measurements in the image. In pure intensity based measurements, the spatial positions of the pixels are not taken into account and all the information is thus retained in the histogram of the image. By intensity, we mean the intensity in a gray-scale image from which the histogram is obtained. The gray-scale image can be obtained using different channels of the original image or by applying different intensity conversions. The color histogram is not used directly as a feature, because it is in our case computed from the whole area of the cell which may contain pixels of the cell, pixels of the parasite, and even pixels with the intensity of the background in the centers of the cells. Moreover, even after the color normalization there is still relatively high variance in color of the parasites due to specific properties of the staining solution used in the particular image.

Intensity and color are the most palpable visual differences between red blood cells and parasites. This difference is a result of the staining procedure during which all the object containing DNA, and thus also the plasmodium parasites, are stained in saturated purple color. In our case, histograms are computed from the whole area of the cell and thus the intensity features may be useful mainly for distinguishing the red blood cells infected by parasites in later stages of development which fill most part of the cell. The intensity features may be advantageous especially when the texture of such a cell is indistinct.

The gray-scale images are obtained from the red, green, blue, saturation and value components of the original image. Although hue information may also be useful, the measurements we use could not be clearly interpreted due to the periodicity of the hue space. The value component represents the case when the actual color is not important. In addition to the original transmission image, histograms are also computed from the extinction image as defined in section 9.2.

Let $h(v)$ denotes the frequency of pixel intensity value v (v = 1, …, N) in the object's histogram h and $p(v) = \frac{h(v)}{A}$ is the probability function which is computed from the histogram by dividing it by the object's area $A = \sum_{v} h(v)$. By calculating moments for the distribution in the histogram, the intensity information can be condensed into a few useful measures. First four central moments are calculated [23]:

Mean

$$\mu = \sum_{v=1}^{N} v p(v) \tag{35}$$

Variance

$$\sigma^2 = \sum_{v=1}^{N} (v - \mu)^2 p(v) \tag{36}$$

Skewness

$$\mu_3 = \frac{1}{\sigma^3} \sum_{v=1}^{N} (v - \mu)^3 p(v) \tag{37}$$

Kurtosis

$$\mu_4 = \frac{1}{\sigma^4} \sum_{v=1}^{N} (v - \mu)^4 p(v) - 3 \tag{38}$$

These are the first four central moments, where the mean gives an estimate of the average

intensity level in the region of the cell and the variance is a measure of the dispersion of region intensity. Histogram skewness is a measure of histogram symmetry and it shows the percentage of the region's pixels that favor intensities on either side of the mean. Kurtosis is a measure of the tail of the histogram. A high kurtosis histogram has a sharper peak and longer tails, while a low kurtosis histogram has a more rounded peak and shorter thinner tail. The subtraction -3 at the end of the Eq.38 ensures that the kurtosis of a Gaussian distribution is normalized to zero.

Additionally, the entropy of the histogram distribution, the seventh largest and smallest intensity value, the median and mode values of the distribution as well as the gray level of the 10th and 90th percentile are also evaluated. The entropy is defined in terms of the histogram as follows [23].

Entropy

$$H = -\sum_{v=1}^{N} p(v) \log_2 p(v) \tag{39}$$

The largest and lowest intensity value in the image obtained from the histogram is usually strongly influenced by noise and, therefore, the seventh largest and smallest intensity values are used instead. Median describes the intensity value separating the higher half of the histogram distribution, mode is the value that occurs the most frequently in the histogram distribution and percentile describes the intensity value below which a certain percent of the histogram distribution falls.

The computation of the intensity features is implemented in function `fHistogram.m`.

9.3.3 Textural Features

Textural features aim to quantify the overall local density variability inside the object of interest. In contrast to the shape and intensity features, textural features are more complex, more difficult to define in a unique, robust and reproducible way, and they are more difficult to understand intuitively. Moreover, it is often difficult to visualize textural features and relate specific feature values to appearance of cells. The difficulty to relate the textural features to visually perceived changes in parasite structure and appearance is the large disadvantage of these features, especially of the co-occurrence and run-length types. However, these features also proved to be one of the most useful ones in many cytological studies [5].

Depending on the development stage and the species of the parasite with which a red blood cell is infected, different types of texture can be observed. Parasites of early development stages form rings with distinct speckles of chromatin. Since the rest of the area of the red blood cell is usually quite intact, the texture is given mainly by the chromatin speckles and possibly by the ring lines which are not always visible (Fig.13a). More distinct texture can be observed in red blood cells infected by parasites in later stages of development. Pigment granules appear early in the growth phase of the parasite as the immature ring-shaped trophozoite becomes mature trophozoite. Depending on the species, the texture in this

phase consists mainly of large amoeboid cytoplasm with large chromatin and fine pigment dots called Schüfner's dots (Fig13b). This is typical mainly for the Plasmodium vivax. In the case of Plasmodium ovale species, trophozoites have sturdy cytoplasm, large chromatin dots and red blood cells are sometimes fimbriated (Fig.12a). In the schizont phase, the nucleus begins to divide, the cell is often filled with the merozoites and the texture is given by large speckles of chromatin, coalesced pigment, and possibly Schüfner's dots, which may also be visible (Fig.13d). Macro- and microgametocytes are characterized by distinct scattered pigment (Fig.13e).

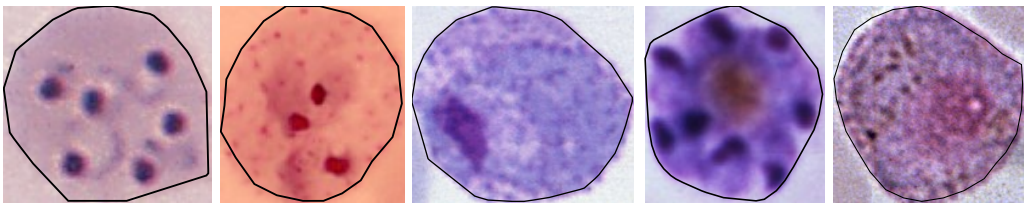


Fig. 13: Different textural properties of infected red blood cells (figures are not to scale). a) P. falciparum ring-form parasites, b) Ring form P. vivax, c) Mature P. vivax trophozoite, d) Immature P. vivax schizont, e) P. vivax microgametocyte

Textural features can be obtained from measurements on certain transformation of the original image. In this work, the following transformations are used: gradient, Laplacian, and flat texture, which are all local operator transformations.

One of the problems with local operations is the dependence of their absolute size on scale. To ensure that the transformations are obtained approximately under the same magnification for all cells, each image with a segmented red blood cell and its corresponding binary mask image are resized with factor 60 / R_A, where R_A is the estimated average cell radius which represents approximately half of the size of a typical red blood cell in an image. An image with a typical non-infected red blood cell with regular shape will thus be resized to approximately 120×120 pixels. The value of 60 pixels was chosen as a typical cell radius in our data set and for data sets with cells of different typical scale this parameter should be changed accordingly.

The transformation could be computed from any of the color channels of the original transmission or extinction image. Calculating the transformation image for all the possible color channels would generate large amount of features with little new information. We assume, that all the texture information is contained in the gray-scale extinction image, from which all the following transformations are computed. This assumption was later justified when we carried out some experiments to compare the performance of certain features for transformations computed from different channels of the original transmission image and the extinction image. For all channels, the extinction image proved to provide better results.

The measurements on the following transformed images are not calculated from the whole area of the cell. In order to remove the border effects in the transformed image, the mask was eroded with disk-shaped structuring element with a fixed size of with a size corresponding to the size of the local operator.

9.3.3.1 Gradient Transformation Features

Gradient of a continuous two-dimensional scalar function $f(x,y)$ is defined as the vector field whose components are the partial derivatives of *f*:

$$\text{grad}(f(x,y)) = \left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right) \tag{40}$$

Derivatives are larger at locations of the image where the image function undergoes rapid changes. The gradient vector in the particular point in the image points in the direction of the greatest rate. The transformation image is obtained from the gradient magnitude which is for a continuous images calculated as:

$$\left|\text{grad}(f(x,y))\right| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x} \right)^2 + \left(\frac{\partial f(x,y)}{\partial y} \right)^2} \tag{41}$$

Since a digital image in discrete in nature, derivatives in Eq.41 must be approximated by differences. The differences of the image *I* in the vertical and horizontal directions are given by

$$\Delta_i I(i,j) = I(i+1,j) - I(i-1,j) \tag{42}$$

$$\Delta_j I(i,j) = I(i,j+1) - I(i,j-1) \tag{43}$$

The differences can be conveniently computed using discrete convolution *G* of image *I* with the convolution mask *H*, which is defined as:

$$G(i,j) = \sum_{(m,n) \in O} \sum H(i-m, j-n) I(i,j) \tag{44}$$

where *O* is the local neighborhood of pixel *I*(*i,j*) in the input image whose pixels are weighted by coefficients *H*.

In our case, *H* is a matrix $[0.7071 \quad 0 \quad -0.7071]$ for horizontal direction and $[0.7071 \quad 0 \quad -0.7071]^T$ for vertical direction. This calculation is equivalent to the computation of differences from Eq.42 and 43, except that the coefficients have value 0.7071 instead of 1 which ensures that the maximum of the gradient magnitude is equal to 1.

Gradient magnitude image is, however, not computed directly from the extinction image I_E.

The image I_E is first filtered by a Gaussian filter to remove local variations in intensity due to noise. The filter is approximated by a 3×3 convolution mask H and the filtered image is computed using Eq.44. The following mask H approximating the Gaussian distribution is used:

$$H = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

The features calculated from the gradient image can be considered as a quantification of the velocity of changes of gray values in the original extinction image. Images containing many objects with well defined edges also contain many pixels with higher values of gradient magnitude. This is valid, for example, for images of infected cell containing several ring-shaped trophozoites. In contrast, a non-infected red blood cell with relatively uniform intensity within its area will typically produce only low values in the gradient magnitude image.

In principal, all the histogram based features computed on the original transmission or extinction image in section 9.3.2, could also be computed using the transformed gradient image. We have selected and further evaluated only the following ones: mean, variance, skewness, and kurtosis (Eq.35-38), histogram minimum and maximum represented by the seventh smallest and largest histogram value, and entropy (Eq.39).

The algorithm for computing features from the gradient magnitude image is implemented in function `fGradientTransform.m` and can be summarized as follows:

1. Rescale both the input extinction image I_E with the cell sample and its corresponding mask image I_M with factor 60 / R_A
2. Filter the rescaled image I_E by Gaussian filter using convolution with Gaussian convolution kernel H_G : I_{E2} = convolution(I_E, H_G)
3. Calculate discrete gradients in the vertical and horizontal direction I_{Gi} and I_{Gj} using Eq.42 and 43. Calculate gradient magnitude image I_G:

   I_G = sqrt(I_{Gi}^2 + I_{Gj}^2)
4. Erode binary mask image I_M using disk-shaped structuring element S of size R_A / 10:

   I_{ME} = erode(I_M, S)
5. Calculate histogram measurements from the gradient values distribution within the region specified by I_{ME}:

   $f_{G\text{-<histogram\_measurement>}}$ = <histogram_measurement>(I_G($p \in I_{ME}$))

R_A is the estimated average cell radius

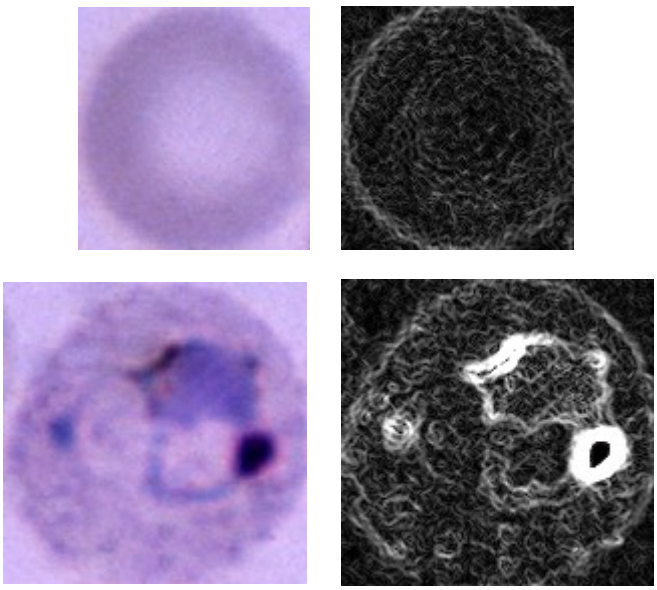


Fig. 14: Gradient images for a non-infected and an infected red blood cells. Original images are shown on the left. Gradient images on the right were calculated from the green channel of the corresponding extinction image. Contrast had to be enhanced and therefore the intensity in the gradient image of the infected cell is saturated.

9.3.3.2 Laplacian Transformation Features

Laplace operator is a differential operator which is defined as the sum of the second partial derivatives. For a continuous two-dimensional scalar function $f(x,y)$, the Laplacian is defined as follows.

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{45}$$

Similarly as in case of the gradient, derivatives in Eq.45 must be in digital images approximated by differences. The discrete approximation of the Laplace operator is computed as:

$$\nabla^2 I(i,j) = \frac{1}{4}\left(I(i+1,j) + I(i-1,j) + I(i,j+1) + I(i,j-1) - 4I(i,j)\right) \tag{46}$$

Laplacian transformation produces large values at locations with large changes of gradient. The resulting features may thus be roughly considered as quantification of the velocity of changes of the gradient.

The transformation image is again not computed directly from the extinction image I_E, but

is first filtered by a Gaussian filter with the same convolution matrix as was used for the gradient transformation.

The same set of seven histogram measurements as in the case of the gradient transformation were calculated from the transformed image.

Laplacian transformation features are evaluated for several sizes of the convolution kernel. The features reflect the intensity of regular particles fitting into the size of the convolution kernel.

The algorithm for computing features from the Laplacian transformation image is implemented in function `fLaplacian.m` and can be summarized as follows:

1. Rescale both the input extinction image I_E with the cell sample and its corresponding mask image I_M with factor 60 / R_A
2. Filter the input extinction image I_E by Gaussian filter using convolution with Gaussian convolution kernel H_G: I_{E2} = convolution(I_E, H_G)
3. Calculate discrete Laplacian transformation using Eq.46:

   I_L = discreteLaplacian(I_{E2})
4. Erode binary image mask I_M using disk-shaped structuring element S of size R_A / 10:

   I_{ME} = erode(I_M, S)
5. Calculate histogram measurements from the image I_L within the region specified by I_{ME}:

   $f_{L\text{-<histogram\_measurement>}}$ = <histogram_measurement>($I_L(p \in I_{ME})$)

R_A is the estimated average cell radius

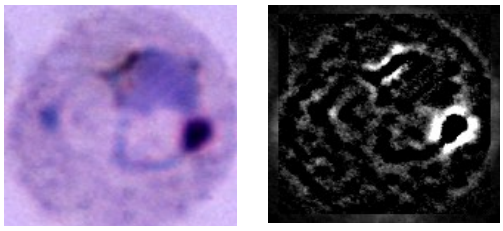


Fig. 15: Laplacian transformation image computed from the green channel of the extinction image (Parameters i,j in Eq.46: i=j=7)

9.3.3.3 Flat Texture

Flat texture is defined as the difference between the original image and a two-dimensional median filtered one [5]. The median is a non-linear operation which is often used to remove noise from images or other signals. It is particularly useful to reduce speckle noise and salt and pepper noise while preserving edges in the image. Median filtering is performed using a window consisting of an odd number of samples. In the median filtered image, the value of each pixel is replaced by median of the values in the pixel's neighborhood which is specified by the window.

Depending on the size of the square median operator window r, the transformation will smooth away particles within the object's region up to an area of half the window area ($r^2/2$). In an infected red blood cell, such smoothed particles may be, for example, the Schüfner's dots or the speckles of chromatin.

Flat texture image I_{FT} is computed directly from the extinction image I_E using the following formula.

$$I_{FT}(x,y) = I_E(x,y) - \mathrm{median}\left(\{I_E(x+\nu, y+\xi);\nu,\xi = -r...r\}\right) \tag{47}$$

where r is the size of the median operator window.

Flat texture image can be considered as a peel of the original image containing only the particles of interest which were smoothed from the original image by the median filtering.

The features computed from the flat texture image, which include the same set of seven histogram measurements introduced in previous sections, are evaluated for different sizes r of the median window and their distinguishing powers are compared. The mask of the cell specifying the region of measurement is reduced by applying morphological erosion with disk-shaped structuring element with radius $r/2$.

The algorithm for computation features from the flat texture image is implemented in function `fFlatTexture.m` and is summarized as follows:

1. Rescale both the input extinction image I_E with the cell sample and its corresponding mask image I_M with factor 60 / R_A
2. Calculate flat texture image I_{FT} from the extinction image I_E using Eq.47
3. Erode binary image mask I_M using disk-shaped structuring element S of size r / 2:

   I_{ME} = erode(I_M, S)
4. Calculate histogram measurements from the image I_{FT} within the region specified by I_{ME}:

   $f_{FT\text{-}\langle\text{histogram\_measurement}\rangle}$ = <histogram_measurement>($I_{FT}(p \in I_{ME})$)

9.3.3.4 Co-occurrence Matrix Features

Co-occurrence matrix is a spatial-dependent matrix representation of the image which estimates the probability that a pixel $I(k,l)$ has intensity i and a pixel $I(m,n)$ has intensity j [7]. Supposing the probability depends only on a certain spatial relation r between a pixel with intensity i and a pixel with intensity j, then the information about the relation r is recorded in the co-occurrence matrix C_r with dimensions corresponding to the number of intensity levels in the image. The spatial relation r can be represented by displacement vector D which is often expressed as distance d and angle θ.

Let $L_X = \{1,2,...,N_X\}$ denote the horizontal spatial domain of the analyzed image with resolution $N_X \times N_Y$, $L_Y = \{1,2,...,N_Y\}$ denote the vertical spatial domain and $G = \{1,2,...,N_G\}$ be the set of N_G quantized gray tones. The input image I is represented as $I : L_Y \times L_X \rightarrow G$. Then the co-occurrence matrix C of dimensions $N_G \times N_G$ for displacement vector $D = [d_1, d_2]$ is defined as originally proposed in [24].

$$C(i,j,D) = \#\{((k,l),(m,n)) \in (L_Y \times L_X) \times (L_Y \times L_X) \mid k - m = d_1, l - n = d_2 \\ I(k,l) = i, I(m,n) = j\} \tag{48}$$

where # denotes the number of elements in the set.

The co-occurrence matrix can be seen as an accumulator matrix to which 1 is added at $C(i,j)$ if a co-occurrence specified by intensities i and j and the spatial relation given by D is found. The co-occurrence matrix defined by Eq.48 is not symmetrical. The symmetrical co-occurrence matrix can be obtained by using absolute values in the distance conditions: $|k - m| = d_1$ and $|l - n| = d_2$. Then the ordering of values in the pixel pairs is not considered and $C(i,j,D) = C(j,i,D)$.

Since the texture in the red blood cell image is directionally homogeneous, we can calculate the co-occurrence matrix using only one displacement vector. However, in order to reduce any irrelevant directional dependencies, we also apply the displacement vector with a rotation of 90° and accumulate the results to the co-occurrence matrix. The following displacement vectors are used: $D = [0, d]$ which corresponds to angle $\theta = 0^\circ$ and $D = [d, 0]$ which corresponds to angle $\theta = 90^\circ$.

The parameters controlling the extraction of the co-occurrence matrix are, in addition to the displacement vector D, also the number of quantized gray levels N_G, which determines the size of the co-occurrence matrix, and the normalization method, which determines how the gray-scale values are scaled to the gray levels. The number of gray levels can, theoretically, be any number. However, for large numbers of levels, the co-occurrence matrix may become sparse with limited generalization properties. The normalization method may be, for example, histogram equalization or linear spread.

The co-occurrence matrix features can be calculated for any of the previously described transformed images as well as for the original extinction image. In our case, co-occurrence matrix is generated from the extinction image I_E and from the flat texture image I_{FT}. The

zone of measurement, i.e. the set of pixels used for generating the co-occurrence matrix, is in case of the extinction image specified by the mask image I_M obtained from the segmentation method, and in case of the flat texture image specified by the eroded mask I_{ME} used for computation of the flat texture features. The following feature set originally proposed in [24] is derived from the co-occurrence matrix. In the following equations, $p(i,j)$ denotes (i,j)th entry in a normalized gray-level co-occurrence matrix, $p_x(i)$ is i^{th} entry in the marginal-probability matrix obtained by summing the rows of $p(i,j)$:

$$p_x(i) = \sum_{j=1}^{N_G} p(i,j) \tag{49}$$

and likewise

$$p_y(j) = \sum_{i=1}^{N_G} p(i,j)\,. \tag{40}$$

Furthermore, $p_{x+y}(k)$ and $p_{x-y}(k)$ are calculated as:

$$p_{x+y}(k) = \sum_{\substack{i=1 \\ i+j=k}}^{N_G} \sum_{j=1}^{N_G} p(i,j)\,, \qquad k = 2, 3, \ldots, 2N_G \tag{51}$$

$$p_{x-y}(k) = \sum_{\substack{i=1 \\ |i-j|=k}}^{N_G} \sum_{j=1}^{N_G} p(i,j)\,, \qquad k = 0, 1, \ldots, N_G - 1 \tag{52}$$

1. *Angular Second Moment*

$$f_1 = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} p(i,j)^2 \tag{53}$$

2. *Contrast*

$$f_2 = \sum_{n=0}^{N_G-1} n^2 p_{x-y}(n) \tag{54}$$

3. *Correlation*

$$f_3 = \frac{\sum_{i=1}^{N_G} \sum_{j=1}^{N_G} (ij)p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{55}$$

where μ_x, μ_y, σ_x, and σ_y are the means and standard deviations of p_x and p_y.

4. *Sum of Squares – Variance*

$$f_4 = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} (i - \mu_x)^2 p(i,j) \tag{56}$$

5. *Inverse Difference moment*

$$f_5 = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} \frac{1}{1 + (i - j)^2} p(i,j) \tag{57}$$

6. *Sum Average*

$$f_6 = \sum_{i=2}^{2N_G} i p_{x+y}(i) \tag{58}$$

7. *Sum Variance*

$$f_7 = \sum_{i=2}^{2N_G} (i - \mu_{x+y})^2 p_{x+y}(i) \tag{59}$$

8. *Sum Entropy*

$$f_8 = -\sum_{i=2}^{2N_G} p_{x+y}(i) \log p_{x+y}(i) \tag{60}$$

9. *Entropy*

$$f_9 = -\sum_{i=1}^{N_G} \sum_{j=1}^{N_G} p(i,j) \log p(i,j) \tag{61}$$

10. *Difference Variance*

$$f_{10} = \sum_{i=0}^{N_G - 1} (i - \mu_{x-y})^2 p_{x-y}(i) \tag{62}$$

11. *Difference Entropy*

$$f_{11} = -\sum_{i=0}^{N_G - 1} p_{x-y}(i) \log p_{x-y}(i) \tag{63}$$

12. *Information Measure of Correlation 1*

$$f_{12} = \frac{HXY - HXY1}{\max(HX, HY)} \tag{64}$$

13. *Information Measure of Correlation 2*

$$f_{13} = \sqrt{1 - e^{-2(HXY2 - HXY)}} \tag{65}$$

where

$$HX = -\sum_{i=1}^{N_G} p_x(i) \log p_x(i) \tag{66}$$

$$HY = -\sum_{j=1}^{N_G} p_y(j) \log p_y(j) \tag{67}$$

$$HXY = -\sum_{i=1}^{N_G} \sum_{j=1}^{N_G} p(i,j) \log p(i,j) \tag{68}$$

are entropies of $p_x(i)$, $p_y(j)$, and $p(i,j)$ and

$$HXY1 = -\sum_{i=1}^{N_G} \sum_{j=1}^{N_G} p(i,j) \log\left(p_x(i) p_y(j)\right) \tag{69}$$

$$HXY2 = -\sum_{i=1}^{N_G} \sum_{j=1}^{N_G} p_x(i) p_y(j) \log\left(p_x(i) p_y(j)\right) \tag{70}$$

We evaluate the co-occurrence matrix features for several lengths of the displacement vector *d*, different number of gray levels N_G and different normalization methods. The details about the specific values of the parameters controlling the extraction of the co-occurrence matrix and discussion on the normalization methods is given in section 9.4.9. Computation of the co-occurrence features is implemented in function `fCooccurrence.m`. The co-occurrence matrix is calculated using Matlab's function `graycomatrix`, which is included in the Image Processing Toolbox. To ensure that the matrix is computed only using the pixel values within the area of the cell defined by the mask image I_M, all pixel values in the input image for which $I_M = 0$ are set to `NaN`. The function ignores pixel pairs if either of the pixels contains a `NaN`.

Function `fCooccurrence.m` can be summarized as follows:

1. Rescale both the input extinction image with normalized intensity levels I_E and the corresponding mask image I_M with factor 60 / R_A
2. Set values of pixels in I_E that are not included in the mask I_M to `NaN`:

   $I_E(p \notin I_M)$ = `NaN`
3. For a given length of the displacement vector *d* and number of gray levels N_G,

calculate sum of the two symmetric co-occurrence matrices for image I_E and for transposed image I_E^T

M_{CO} = graycomatrix(I_E) + graycomatrix(I_E^T)

4. From normalize matrix $p_{M_{CO}} = \frac{M_{CO}}{\sum_{i=1}^{N_G}\sum_{j=1}^{N_G} M_{CO}(i,j)}$ calculate features given by Eq.53 – 65

9.3.3.5 Run-length Matrix

Run-length statistics is capable of capturing the coarseness of a texture in specified direction. A run is defined as a string of consecutive pixels which have the same gray-level intensity along a specific linear orientation [25]. In a coarse texture, relatively long gray-level runs with significantly different intensities can be observed more often while a fine textures tend to contain primarily short runs with similar gray-level intensities.

For a given image, a run-length matrix P is defined as a matrix in which each element $P(i,j)$ represents the number of runs with pixels of gray-level intensity equal to i and length of run equal to j along a specific orientation. The dimension of the matrix is M by N, where M is the number of gray levels in the quantized image and N is the possible maximum run length. An orientation is defined using a displacement vector $D(x,y)$. For typical orientations 0°, 45°, 90°, and 135°, the following representations of D can be used: (1,0), (1,1), (0,1), (–1,1).

Since the texture in the red blood cell image is directionally homogeneous, we can calculate the run-length matrix using only one displacement vector. However, similarly as in the case of the co-occurrence matrix, we also apply the displacement vector with a rotation of 90° and accumulate the results to the run-length matrix in order to reduce any irrelevant directional dependencies.

The parameters controlling the generation of run-length matrix are the displacement vector D specifying the direction in which the runs are calculated, the number of quantized gray levels N_G, which determines the width of the matrix, the maximum considered run length N_R, which determines the height of the matrix, and the normalization method.

The run-length matrix is generated from the the area of the extinction image I_E specified by the mask image I_M and from the region in the flat texture image I_{FT} specified by the eroded mask I_{ME} used for computation of the flat texture features. Eleven features proposed in [26] are calculated from the generated run-length matrices and evaluated. The feature generating measurements on the run-length matrix are defined as follows.

1. *Short Run Emphasis (SRE):*

$$\mathrm{SRE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)}{j^2} \tag{71}$$

2. *Long Run Emphasis (LRE):*

$$\mathrm{LRE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}p(i,j)\cdot j^2 \tag{72}$$

3. *Gray-Level Nonuniformity (GLN):*

$$\mathrm{GLN} = \frac{1}{n_r}\sum_{i=1}^{M}\left(\sum_{j=1}^{N}p(i,j)\right)^2 \tag{73}$$

4. *Run Length Nonuniformity (RLN)*

$$\mathrm{RLN} = \frac{1}{n_r}\sum_{j=1}^{N}\left(\sum_{i=1}^{M}p(i,j)\right)^2 \tag{74}$$

5. *Run Percentage (RP)*

$$\mathrm{RP} = \frac{n_r}{n_p} \tag{75}$$

6. *Low Gray-Level Run Emphasis (LGRE)*

$$\mathrm{LGRE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)}{i^2} \tag{76}$$

7. *High Gray-Level Run Emphasis (HGRE)*

$$\mathrm{HGRE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}p(i,j)\cdot i^2 \tag{77}$$

8. *Short Run Low Gray-Level Emphasis (SRLGE)*

$$\mathrm{SRLGE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)}{i^2\cdot j^2} \tag{78}$$

9. *Short Run High Gray-Level Emphasis (SRHGE)*

$$\mathrm{SRHGE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)\cdot i^2}{j^2} \tag{79}$$

10. Long Run Low Gray-Level Emphasis (LRLGE)

$$\text{LRLGE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)\cdot j^2}{i^2} \tag{80}$$

11. Long Run High Gray-Level Emphasis (LRHGE)

$$\text{LRHGE} = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}p(i,j)\cdot i^2\cdot j^2 \tag{81}$$

In the equations above, n_r is the total number of runs and n_p is the number of pixels in the image.

We compute the run-length matrix features for several different numbers of gray levels N_G and different normalization methods. The details about the parameters used are given in section 9.4.10. Computation of the run-length features is implemented in function `fRunLength.m`, which can be summarized as follows.

1. Rescale both the input extinction image with normalized intensity levels I_E and the corresponding mask image I_M with factor 60 / R_A
2. Scale image to N_G gray levels
3. Calculate run-length matrices for image I_E and for transposed image I_E^T

   RLE_1 = rle(I_E)

   RLE_2 = rle(I_E^T)

4. Let [M,N_1] be the size of the matrix RLE_1 and [M,N_2] be the size of the matrix RLE_2.

   Accumulate results from matrices RLE_1 and RLE_2 into a new matrix RLE with size [M,min(max(N_1,N_2),N_R], where N_R is a given parameter specifying the maximum run length.

5. Calculate features according to Eq.71 – 81

The computation of the run-length matrix is implemented in the function `rlm.m`. The algorithm can be described as follows.

```
FUNCTION rlm(input_image, mask, number_of_levels)

SET input_image to 0 at positions where mask = 0
CREATE new zero array run_length_matrix of size
      (number of columns in input_image, number_of_levels)

FOR each row img_row in input_image
      ind = indices of intensity changes in img_row
      lengths = differences between adjacent elements of ind
      values = img_row at positions (ind)
      ind_mask = indices of values where values > 0
      values_mask = values at positions (ind_mask)
      lengths_mask = lengths at positions (ind_mask)
      FOR each pair in (values_mask, lengths_mask)
            INCREMENT run_length_matrix at positions (values_mask,
      lengths_mask)
      END
END

RETURN run_length_matrix
```

9.4 Feature Selection

Features proposed in the previous section are generated using different transformation parameters and further evaluated in order to select the ones with best discrimination power. The generated features are evaluated in two aspects. First, the discriminating capabilities are analyzed individually for each feature using the receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC). The combinations of best performing features are then evaluated using criterion based on scatter matrices in order to propose feature vectors with higher discrimination power.

9.4.1 The Receiver Operating Characteristic Curve

Fig.16a illustrates an example of two overlapping probability density functions describing the distribution of feature values for two classes, which are in our case an infected and non-infected red blood cell. For the sake of generalization, let us call the infected red blood cell as positive (p) class and the non-infected red blood cell as negative (n) class, which fully conforms to the actual meaning of the classes. By choosing a threshold, we can decide class n for values on the left of the threshold and class p for the values on the right. This decision is associated with an error probability, FPR, of reaching a wrong decision concerning the class p. This probability is called as false positive rate. The probability of a correct decision is $TPR = 1 - FNR$, called as true positive rate. Similarly, FNR (false

negative rate) is the probability of a wrong decision concerning class *n* and *TNR* = 1 – *FPR* (true negative rate) is the probability of a correct decision concerning class *n*. By moving the threshold over the whole range of possible positions, we obtain different values of *FPR* and *FNR*. The ROC curve is defined as a plot of *TPR* as the y coordinate versus *FPR* as the x coordinate [18,27]. If the two distributions have complete overlap, then for any position of the threshold: *TPR* = 1 – *FPR*. Such a case corresponds to the straight line in Fig.16b. The feature producing such ROC curve has no discrimination capability. As the two distributions move apart, the corresponding curve departs form the straight line and the less the overlap of the classes, the larger the area between the curve and straight line. Two completely separated class distributions would result in *TPR* = 1 for all the threshold positions, i.e. for the whole interval [0,1] of *FPR*. Such ROC curve would be produced by a feature with ideal discriminating capability. Thus the area under the ROC curve (AUC) is a measure of the class discrimination capability of the specific feature. The simplest way to calculate this area is to use trapezoidal integration using the following formula [28], in which α represents the *FPR* and β represents the *FNR* (i.e. $1-\beta = TPR$).

$$\mathrm{AUC} = \sum_i \left(\left((1-\beta_i) \cdot \Delta\alpha \right) + \frac{1}{2} \left(\Delta(1-\beta) \cdot \Delta\alpha \right) \right) \tag{82}$$

where,

$$\Delta(1-\beta) = (1-\beta_i) - (1-\beta_{i-1}) \tag{83}$$

$$\Delta\alpha = \alpha_i - \alpha_{i-1} \tag{84}$$

The ROC curve is constructed by sweeping the threshold and computing percentages of wrong and correct classifications for the particular feature, i.e. the false positive rate and the true positive rate. If a simple binary classifier were to be devised based on the particular feature by choosing a threshold value, this value would correspond to one point on the ROC curve. If two classifiers with different thresholds were to be compared, the better classifier is the one producing smaller number of false positives and at the same time greater number of true positives, i.e. the better classifier is represented by a point to the north and to the west relative to the point of the worse classifier.

Computation of ROC curve points and the corresponding AUC is implemented in function `roccurve.m`. Function `plotroccurve.m` is used to plot the calculated curves.

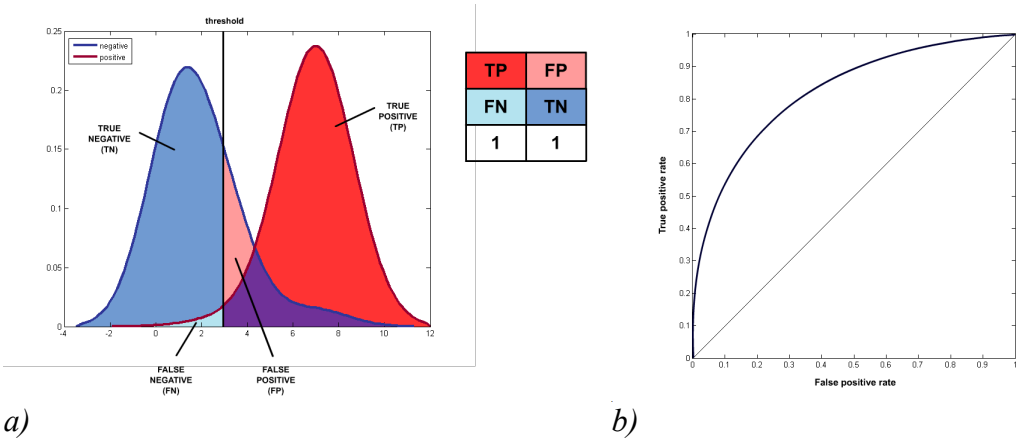


Fig. 16: Probability density functions for positive and negative class a). ROC curve b).

9.4.2 Feature Evaluation m-scripts

The features are evaluated in groups according to the transformation from which they are calculated. To calculate the features using all input samples and visualize the results, the following Matlab scripts have been written.

`evalHu.m` – evaluates Hu's set of invariant image moments

`evalShape.m` – evaluates the relative shape measurements

`evalHist.m` – evaluates the intensity features based on image histogram

`evalGrad.m` – evaluates gradient transformation features

`evalLaplacian.m` – evaluates Laplacian transformation features

`evalFlatTexture.m` – evaluates flat texture features

`evalCooccurrence.m` – evaluates co-occurrence matrix features

`evalRunLength.m` – evaluates run-length features

These Matlab m-files are not functions and so they have no input or output arguments. They are intended for evaluation and visualization of the results and they are to be edited to deliver specific output, if required. Each of these scripts is programmed to carry out the following actions.

1. Specifying the directory containing input image files with corresponding .mat files with the information about the segmented cell contours which together form the input red blood cell database. The directory is, by default, set to `'..\..\rbcSegm\images\highres3'`, but the script also includes a code to open standard dialog box for selecting a directory. This code can be simply commented out, if required.

2. A list of all image files in the directory is retrieved.

3. A waitbar figure is initialized. The waitbar shows a progress bar and exact percentage of input red blood cell samples already processed. It also contains a cancel button which enables interruption of the feature calculation process at any time.

4. The input red blood cell samples are processed within two loops. The first loop successively loads all image files listed in the previously retrieved file list. The structure array containing the segmented red blood cell contours' information is loaded for each image file from the corresponding mat-file. If the mat-file does not exist the program continues with the next iteration of the loop and loads the next image file from the list. If the database file exists, specific preprocessing steps and image transformations are carried out. This includes actions that have to be performed on the whole original image, such as non-uniform illumination correction or color normalization.

5. The nested second loop iterates through all red blood cell samples in the particular image. According to the information in the loaded database structure array, an image containing only a single red blood cell sample is generated together with a mask image specifying the zone of measurement. The mask image is computed as the area specified by the segmented red blood cell contour. Within this loop, features are calculated for the particular image transformation using a set of input parameters or using various versions of the input image, on which the transformation is calculated. The results are stored in one multidimensional array. As a rule, the first dimension corresponds to the individual red blood cell samples and its length equals to the total number of available samples in all input images. Additionally, one-dimensional array called `targets` is created containing the information about the class of the sample. Its length is equal to the total number of red blood cell samples and it contains zeros on positions corresponding to the non-infected red blood cells (the negative class) and ones on positions corresponding to the red blood cells infected by malaria (the positive class).

6. Since the process of feature calculation for different input parameters may be time consuming, both the multidimensional array with feature values and the `targets` array are stored in a mat-file with the same name as is the name of the script (e.g. `evalHu.mat`). The variable can be loaded at any time so that any evaluation of the results, such as plotting the ROC curves, can be easily carried out later.

7. The computed features are evaluated by calculating the ROC curve and the area under the curve (AUC) for each feature. ROC curves and estimated probability

density functions are plotted for selected features depending on the particular feature set.

The ROC curve together with AUC are calculated using the `roccurve.m` function, which is described in section 9.4.1. The ROC curves are plotted using `plotroccurve.m` function, which generates multiple plots in different colors in a square axes region together with a diagonal line between points [0,0] and [1,1]. Title and legend can also be specified.

The estimated probability density functions of the feature values for infected and non-infected red blood cells are plotted using function `plotFeature.m`. The probability density functions can be estimated using a normalized histogram. However, in order to obtain a smooth curve, the density estimate is computed using Matlab function `ksdensity`, which uses normal kernel function to smooth the resulting curve.

By selecting another directory, altering the current red blood cell database or adding new segmented images to the database, the performance of the features can be easily evaluated for new sets of red blood cells using these scripts. Moreover, a new data set can be easily created using the segmentation GUI containing only non-infected red blood cells and red blood cells infected by malaria parasite of specific species or in a given stage of development. The scripts may then be used without any change to evaluate the features on a particular subset of malaria parasites, as shown in the next section. However, the scripts are intended only for evaluation of the features on two classes.

9.4.3 Hu Set of Invariant Moment Features

Since these features are calculated only from the mask of the red blood cell, no image preprocessing or image transformations are needed. In fact, even loading the original image file is not required. The script for evaluation the invariant moment features is stored in `evalHu.m` file. The feature values are stored in a two-dimensional $N \times F$ array `I`, where N is the total number of samples and F=7 is the number of calculated features for each red blood cell sample image. Array `I` together with the variable `targets` indicating the class of the red blood cell are saved in `evalHu.mat` file.

The estimated probability density functions, which are depicted in Fig.17, indicate that the discriminating power of the features evaluated for the set containing red blood cells infected with all kinds of malaria parasite will be very low. This was partly expected because most of the infected red blood cells in the data set contain parasites in early stages of development and these cells show no significant difference in shape from the non-infected cells. The ROC curves for all seven Hu's invariant moments are shown in Fig.18 and the AUCs calculated from the corresponding ROC curves are in Tab.5. The highest AUC was acquired for the third Hu's moment, the difference between the values of the individual moments is, however, insignificant and the performance of all the feature can be concluded as poor.

	H1	H2	H3	H4	H5	H6	H7
AUC	0.5284	0.5285	0.5500	0.5232	0.5053	0.5102	0.5224

Tab. 5: Area under the ROC curve for Hu set of invariant moment features

In order to find out whether these features could possibly be useful for a certain subset of infected red blood cells, a new data set was created using the developed GUI tool. Specifically, this data set contains only red blood cells infected by mature plasmodium falciparum parasites which have characteristic crescent-like shape. The new dataset was created by copying all image and mat-files forming the current database to a new directory called 'highres3-falciparum'. The contours of the segmented red blood cells in the individual images were then edited by the segmentation GUI tool and all infected red blood cells that were not infected by mature plasmodium falciparum parasite were deleted from the database. Any other subset containing only cells infected by a specified type of plasmodium parasites can be created in the same way. The array with the calculated invariant moment features and the target array is saved for later use in `evalHu-falciparum.mat` file. The estimated probability density functions are shown in Fig.21, the corresponding ROC curves for the first four Hu moments are displayed in Fig.19 and the calculated AUCs are in Tab.6. The main problem associated with the data set containing only Falciparum parasites in the positive class was the small number of available samples in the positive class. This prevents us from drawing any statistically significant conclusions. However, the very high values of AUCs (AUC_{Hu1} = 0.9998, AUC_{Hu2} = 0.9999) and very good separation of classes for the first two Hu's moments indicate that these features could indeed be useful in certain specific cases. The main reason why the performance of the invariant moment features was inferior when all types of parasites were included in the data set, was caused by the fact that most of the infected red blood cells in the data set were not significantly distinguishable from the rest of the cells based only on their shapes and the number of samples with specific shape, such as the P. falciparum infected cells, was very low.

	H1	H2	H3	H4	H5	H6	H7
AUC	0.9998	0.9999	0.9969	0.9982	0.8740	0.8743	0.4952

Tab. 6: Area under the ROC curve for Hu set of invariant moment features calculated using non-infected red blood cells and red blood cells infected by mature Plasmodium Falciparum parasites

9.4.4 Relative Shape Measurement Features

The second group of features that are based only on the actual mask of the segmented red blood cells are relative shape measurements. The five calculated measurements include relative area of the cell, relative perimeter, relative lengths of the major and minor axes of the best-fit ellipse and the relative radius of the largest inscribable circle. The performance of this group of features is evaluated in the script `evalShape.m` and the generated two-dimensional array `S` is saved in file `evalShape.mat`. The first dimension of the array corresponds to the individual red blood cell samples and the second array of length five corresponds to the individual features.

Similarly as in the case of the invariant moments, the shape features are calculated only from the mask of the segmented red blood cell and, therefore, no image preprocessing or image transformations are necessary.

The estimated probability density functions for the shape feature values are depicted in Fig.22, the corresponding ROC curves are shown in Fig.20 and the calculated AUCs are in Tab.7. The results reflect the types of infected red blood cell shapes in the data set. As stated in the previous section, most of the infected red blood cells in the database contain parasites in early stage of development and such cells differ insignificantly from non-infected cells. Relatively smaller number of infected cells are enlarged or elongated compared to non-infected cells whose size is represented by the estimated red blood cell radius against which the individual shape measurements are normalized. Only a small number of infected red blood cells have specific shape such as the cells infected by Plasmodium Falciparum or some cells which are nearly ruptured due to the growing parasite. For this reason, the performance of the features calculated only from the mask image is necessarily limited. Compared to the invariant moment features, which are scale invariant, shape measurements can detect changes in size. This might explain the slightly better performance of these features compared to the invariant moments.

	Area	Perimeter	Major axis of best-fit ellipse	Minor axis of best-fit ellipse	Largest inscribable circle
AUC	0.6162	0.6382	0.6462	0.6023	0.5861

Tab. 7: AUCs for relative shape measurement features

As can be seen from Fig.22, the probability density functions for the positive and negative class show extensive overlap for all calculated features. Moreover, the functions exhibit similar behavior for all the features, which is in accordance with what we would expect considering the distribution of the shapes in our data set. Although the individual measurements describe different aspects of the object's shape, they are all approximately proportional to the size of the object for highly regular shapes, such as circles are. The

distributions show that all the measurement values for the negative class are approximately normally distributed around one for the area, perimeter and largest inscribable circle and around two for the best fit ellipse (this is because the lengths of the minor and major axes of the best-fit ellipse are normalized by the average cell radius and not the diameter). The mean radius of the largest inscribable circle is slightly less then one because any difference of the shape from a circle will result in smaller inscribed circle. This happens, for instance, when a cell is partially cut as a result of separating it from a compound during segmentation. The distributions of the shape measurement values for the infected red blood cells are similar, but shows a longer right tail in all cases. This tail represents measurements on the enlarged or elongated infected red cells.

The largest value of AUC was obtained for the length of the best-fit ellipse's major axis while the measurement of the largest inscribable circle radius resulted in the smallest value of AUC. The differences between the values obtained for the individual measurements are, however, not significant and, generally, the discrimination power of all these features was rather inferior.

Similarly as in the case of the invariant moment features, the shape measurements could be evaluated on a certain subset of the infected red blood cells; this evaluation is, however, not performed in this work. Possibly, they could be found useful for detecting of red blood cells infected by plasmodium parasites in a certain (later) stage of development. However, the low values of AUC and the similarity of the individual estimated density functions suggest that, for a general case of distinguishing between infected and non-infected red blood cells, the usability of these features can be considered as low.

Generally, the advantage of the shape measurements is easy and fast computation and rather straightforward interpretation of the calculated feature values. The disadvantage is certain dependence of the shape on the preceding segmentation method which can be observed mainly in cells separated from compounds and in cells with rugged or crenelated contours. For our problem of infected and non-infected red blood cell classification, the general disadvantage of features based only on the mask of the segmented cell is the low amount of information that is captured solely in the shape of the object.

9.4.5 Histogram Features

This set of intensity features is based only on the absolute values of pixel intensities without taking the spatial positions of the pixels into account. All the information for computation of these features is thus retained in the histogram of the image.

Since these features are based only on the absolute intensity values, it is crucial to ensure that the same objects are represented with the same intensities within the whole area of an image as well as in all input images. Therefore, two preprocessing steps are first performed in each original image – correction of non-uniform illumination and color normalization. Additionally, each input image shows different contrast. In some images, red blood cells are shown in pale colors and parasites in dark colors while in other images, the difference in intensity between red blood cells and parasites is much less pronounced. Ideally, for

each image a contrast correction function should be evaluated in order to normalize the contrast across all the input images. However, in practice this is not a trivial task because images contain different types and numbers of parasites and their required intensity may be unknown. Differences in contrast are, therefore, not corrected and must be accepted as the inevitable variance in input parameters. In real-life applications, the contrast in the image could be controlled in the input device so that the contrast normalization would not be required. In addition to the transmission image, the extinction image is also computed from the preprocessed original image and the histogram features are calculated from both images. The zone of measurement specifying pixels from which the histogram measurements are calculated is the whole area of the cell, which is specified by its mask.

A vector of eleven intensity measurements is calculated from the histogram of each R, G, and B channels of the original and extinction images and also from the the S and V channels of the images transformed to the HSV color space. Calculating statistical measurements from the histogram of the hue channel makes no sense because of the periodicity of the hue space. Evaluation of the histogram-based features is implemented in the script `evalHist.m` and the three-dimensional array `H` containing the calculated features is saved for later use in file `evalHist.mat`. The first dimension of this array corresponds to the individual red blood cell samples and the second dimension corresponds to the channels R_O, G_O, B_O, S_O, V_O, R_E, G_E, B_E, S_E, and V_E, where O denotes the preprocessed original image and E denotes the extinction image. The third dimension of the array corresponds to the eleven calculated features. The evaluated features include mean, variance, skewness, kurtosis, the seventh minimum, the seventh maximum, median, mode, 10^{th} percentile, 90^{th} percentile, and entropy.

	R_O	G_O	B_O	S_O	V_O	R_E	G_E	B_E	S_E	V_E
Mean	0.6696	0.8172	0.6369	0.6595	0.7741	0.7743	0.8372	0.7043	0.5019	0.8241
Variance	0.8909	0.9379	0.8522	0.9594	0.8757	0.9378	0.9740	0.8950	0.4881	0.9732
Skewness	0.9140	0.8917	0.9531	0.7992	0.8993	0.9486	0.9560	0.9752	0.5139	0.9419
Kurtosis	0.7925	0.7879	0.8255	0.7920	0.8033	0.8749	0.8485	0.9033	0.5846	0.8459
Min.	0.9712	0.9962	0.9593	0.4965	0.9908	0.6926	0.6136	0.6956	0.5335	0.6176
Max.	0.5272	0.5567	0.5517	0.9315	0.5718	0.9721	0.9959	0.9570	0.5197	0.9954
Median	0.6367	0.7773	0.6080	0.5966	0.7428	0.7188	0.7808	0.6615	0.5021	0.7690
Mode	0.6841	0.7628	0.6322	0.6327	0.7188	0.6836	0.7304	0.6499	0.5240	0.7397
10^{th} percentile	0.7280	0.8347	0.6644	0.4172	0.7924	0.6688	0.7489	0.6323	0.4962	0.7226
90^{th} percentile	0.5837	0.7308	0.5947	0.6761	0.7069	0.7948	0.8337	0.7281	0.5010	0.8245
Entropy	0.8170	0.8532	0.7347	0.8279	0.8004	0.8462	0.8704	0.7528	0.4857	0.8615

Tab. 8: Area under the ROC curve for histogram-based features

The calculated AUCs for all feature values are shown in Tab.8. The field with the best result (highest AUC) for each feature is highlighted in yellow color. It can be seen from the

table that most of the best results were obtained for the green channel of the extinction image, followed by the green channel of the original transmission image and the blue channel of the extinction image. If the individual features are ordered according to the highest achieved value of AUC from the highest to the lowest value, we will obtain:

> Minimum, maximum, skewness, variance, kurtosis, entropy, mean, 10^{th} percentile, 90^{th} percentile, median, and mode.

The first two measures, minimum and maximum, are complementary, because minimum in the green channel of the transmission image corresponds to the maximum in the green channel of the extinction image. This holds true also for the red, blue and value channels and it is the result of the transformation to the extinction image. Indeed, the AUC value for the histogram minimum in the the green channel of the transmission image is almost the same as the AUC value for the histogram maximum in the green channel of the extinction image. Equally large value of AUC was also achieved for the maximum in the value channel of the extinction image followed by the minimum in the value channel of the transmission image. These are the only four measurements whose AUC is greater than 0.99. Due to the high mutual correlation between the individual channels, it is no surprise that high values of AUC were also achieved for the minimum in the R, B, and V channels of the transmission image and for the maximum in the the same channels of the extinction image.

Other measurements with large AUC values are the variance of the green channel and the skewness of the blue channel in the extinction image. The performance of all the other measurements is significantly lower.

The ROC curves for all measurements are depicted in Fig.25, 26, 27, and 28 and the estimated probability density functions for all measurements in the green channel of the extinction image are shown in Fig.23 and 24. The ROC curves demonstrate that, generally, the most useful set of measurements calculated from the gray-scale histograms include variance, skewness, kurtosis, and minimum or maximum, followed by entropy. The other measurements including mean, median and percentiles did not prove to deliver acceptable results and should not be used. The ROC curves also show that measurements on the saturation channel do not deliver, in most cases, acceptable results. The only exception was the histogram maximum of the saturation channel in the transmission image. Even in this case, however, the ROC curve is rather asymmetrical, which is the result of the bimodal character of the underlying distribution functions for both classes. It is not surprising that the best results were achieved with the measurements calculated on the green channels of both images. This is due to the fact that parasites are typically shown in dark saturated purple color and, therefore, in the green channel the highest contrast between pixels of parasites and pixels of red blood cells, which are usually shown in lighter red colors, is achieved. It is, however, important to remark that these result are dependent on the channel constants used during the color normalization process. If color normalization is not performed or is not well controlled, it is more appropriate to use the value channel of the extinction image. The values of AUCs for most of the features were almost as high for this channel as for the green channel of the extinction image and, on average, the second highest, although this channel did not deliver the best results for any of the measurements.

The evaluation of histogram-based features has shown that the best discrimination power can be, according the the AUC measure, provided by the histogram maximum in the green channel of the extinction image, or alternatively the maximum in the value channel of the extinction image or the minimum of the green in the original image. Although the images in our data set exhibit relatively high amount of noise which can affect the detected maximum and minimum intensity values in the image, choosing the seventh maximum and minimum value in the histogram proved to provide good results. More experiments could be carried out to evaluate whether another value could possibly deliver better results.

	G_E	$G_{E(\text{median-filtered})}$	$G_{E(\text{eroded mask})}$
Variance	0.9740	0.9692	0.9733
Skewness	0.9560	0.9442	0.9623
Kurtosis	0.8485	0.8393	0.8529
Minimum	0.6136	0.7063	0.6574
Maximum	0.9959	0.9968	0.9967
Entropy	0.8704	0.8502	0.8725

Tab. 9: Comparison of the AUC values for selected histogram measurements calculated from the green channel of three versions of the extinction image: G_E – green channel of the extinction image without any further processing; $G_{E(\text{median-filtered})}$ – green channel after applying median filter; $G_{E(\text{eroded mask})}$ – green channel without any processing, but with the zone of measurement specified by an eroded mask

We were also interested whether the results would improve in case the noise in the green channel of the extinction image was suppressed by applying a median filter and in case the zone of measurement was reduced by eroding the cell mask. The reduction of the area of the image from which the histogram is obtained was motivated by the fact that the cell margin may contain brighter or darker pixels or that the mask may be inaccurate and thus the specified area may contain pixels of the background. The mask was eroded using disk-shaped structuring element with radius R_A / 8, where R_A is the estimated average radius of the cells in the image. The computed AUCs for these two additional scenarios are shown in Tab.9 together with the AUCs calculated from the original green channel of the extinction image for easy comparison. Only variance, skewness, kurtosis, minimum, maximum and entropy were evaluated. The results show that after applying the median filter, the performance of all measurements deteriorates slightly, except for the histogram minimum and maximum. On the other hand, after removing pixels from the marginal regions of the cell, the performance improves slightly for all measurements except the variance. The highest AUC achieved increased from 0.9959 to 0.9968. This experiment shows that median filtering may remove some information useful for discrimination between classes while reduction of the measurement area may slightly improve the performance by selecting cell pixels with higher relevance to class differentiation.

In general, the advantage of the histogram-based features is the easy and fast computation. Compared to the textural features, intensity features can also be more easily interpreted and linked to the observable image properties. The disadvantage of these features is that they are based on the absolute intensity values of the pixels and, therefore, shading correction, color and possibly also contrast normalization are of great importance.

9.4.6 Gradient Transformation Features

Since gradient is not dependent on the absolute value of the overall pixel intensity within the cell area, the correction of non-uniform illumination is not required. The gray-scale extinction image, from which the gradient features are calculated, is obtained from the green channel of the original image. The measurements calculated from the gradient transformation image include mean, variance, skewness, kurtosis, the seventh maximum and minimum in the histogram and the entropy of the histogram. Evaluation of the gradient transformation features is implemented in the script `evalGrad.m` and the generated two-dimensional array `G` containing the calculated feature values is saved in the mat file with the same name.

The gradient transformation was computed using the default parameters as described in section 9.3.3.1, i.e. the image is first filtered by a Gaussian filter with 3×3 convolution mask and the gradient transformation is computed using convolution mask of length 3.

The calculated AUCs (see Tab.10) indicate that the best distinguishing power can be obtained by calculating the variance of the gradient image histogram. Relatively high values of AUC were also obtained for the seventh largest and smallest value in the histogram. It can be seen from the corresponding ROC curves (see Fig.29) that although the calculated AUC value for the histogram entropy is not significantly smaller than the value for histogram minimum, the ROC curve for the entropy is much flatter compared to the ROC curve of the histogram minimum so that its points are located further from the point [0,1]. For this reason, the real performance of a classifier with such feature can be expected to be even lower.

	Mean	Variance	Skewness	Kurtosis	Minimum	Maximum	Entropy
AUC	0.9029	0.9943	0.9112	0.8781	0.9665	0.9866	0.9450

Tab. 10: AUCs for gradient transformation features using the default size of the Gaussian filter H_G=[3,3] and length of the gradient convolution kernel l_G=3

	H_G=[3,3], l_G=3	H_G=[3,3], l_G=5	H_G=[7,7], l_G=3	H_G=[7,7], l_G=5
Extinction image	0.9943	0.9971	0.9943	0.9971
Median filtered ext. im.	0.9973	0.9977	0.9973	0.9977

Tab. 11: AUCs for variance of the gradient image, where H_G denotes the size of the Gaussian filter and l_G denotes the length of the convolution kernel used for computation of the image gradient

Additional experiments were carried out to evaluate how the performance depends on the change of certain transformation parameters. Specifically, we calculated AUC of the variance of the gradient image histogram in these cases:

- The extinction image is first filtered by a median filter with 3×3 window before the gradient transformation is calculated
- The image is filtered by Gaussian filter with convolution matrix 7×7 instead of the default size 3×3
- The gradient is calculated using convolution mask with length 5 instead of default length 3 ($H = [0.5 \quad 0 \quad 0 \quad 0 \quad 0.5]$ for horizontal direction and $[0.5 \quad 0 \quad 0 \quad 0 \quad 0.5]^T$ for vertical direction)

Median filter is a non-linear filter that is known to be capable of reducing noise while preserving edges in the image. Although it proved to decrease the discriminating power of some histogram-based measures on the original extinction image, it could possibly improve the results in this case, because the gradient image is rather highly sensitive to noise. Gaussian filtering is a standard operation performed to reduce noise in the image before the gradient is computed. It smooths noise and also the edges in the images reducing random high variations in gradient. The evaluation of a different length of the convolution kernel used for calculation of the gradient was motivated by the fact that the width of the parasite edges in the resized red blood cell image could be greater than three pixels and thus possibly better detected by this convolution kernel. The resulting AUCs are shown in Tab.11. It demonstrates that median filtering may, indeed, slightly improve the performance of the histogram variance feature, as well as increasing the length of the convolution kernel for calculation of the gradient. On the other hand, changing the size of the Gaussian filter resulted in no change in the corresponding ROC curve for this particular measurement.

In conclusion, we have shown that variance of the pixel values in the gradient image obtained from the green channel of the extinction image can be useful in distinguishing between infected and non-infected red blood cells. Median filter may improve the results if the images contain significant amount of noise and the convolution mask for calculating the gradient can be adjusted to reflect the real size of the image and to deliver the optimal results.

9.4.7 Laplacian Transformation Features

Similarly as in the case of gradient transformation features, Laplace operator is independent of the overall intensity level within the cell area and, therefore, the correction of the non-uniform illumination in the input image is not strictly required. However, better results were observed when the correction was applied. This is in contrast with gradient transformation features where only insignificant changes were observed when the non-uniform illumination correction was performed. Although both these transformations are based on intensity changes in the image, in this case the measurements on the transformation image are evaluated for greater sizes of the convolution mask which covers an area of the cell large enough to be affected by the shading correction, while the gradient was calculated only locally for the size of the convolution mask smaller than five pixels. We also observed that the performance of the evaluated features was generally higher when the extinction image was first filtered by the median filter. The only exception was the kurtosis measurement on the histogram of the transformed image which showed lower performance after the median filter was applied. This measurement, however, did not prove to deliver results superior to the other measurements even for non-filtered extinction image. For these reasons, both non-uniform illumination correction and median filtering is applied on the extinction image from which the Laplacian transformation image is computed for all features. The extinction image is obtained from the green channel of the original transmission image.

The Laplacian transformation features are evaluated in the script `evalLaplacian.m` and a four-dimensional array `L` containing the values of the calculated features is saved for later use in the mat-file with the same name. The first dimension in this array corresponds to the individual red blood cell samples and the second dimension of length 2 corresponds to the non-filtered and filtered extinction image from which the Laplacian transformation is computed. The features are evaluated for different values of the parameter r in Eq.46, which controls the size h of the convolution matrix ($h = 2r + 1$). Feature values are calculated for the following parameters of $r = \{1,3,5,7,9,15\}$ using seven measurements on the histogram of the transformed image, which include mean, variance, skewness, kurtosis, minimum and maximum represented by the seventh smallest and largest histogram values, and entropy. The last two dimensions of the array `L` correspond to the parameters r and individual listed measurements.

The values of calculated AUCs are shown in Tab.12. It can be seen from the results that the only useful measurements include the variance and the maximum. The highest value were for both of these measurements achieved for $r = 7$. Indeed, also the values of AUC for other measurement indicate that most information useful for distinguishing non-infected and infected red blood cells is contained in the Laplacian transformation image obtained for the value of the parameter r between 7 and 9. It has to be emphasized that the value of this parameter is only relevant in relation to the red blood cell size. In our case, all red blood cell images are resized by the factor of $R_A / 60$, where R_A is the estimated radius of a typical non-infected red blood cell in the image. In this context, the value of $r = 7$ represents objects within a red red blood with the size of approximately $R_A / 4$. After examination of the input images, we can see that there are indeed many objects with approximately this size. These objects are the speckles of chromatin in the ring-shaped

parasites. The estimated probability density functions for the variance and maximum of the Laplacian transformation image histogram are shown in Fig.30 for $r = 7$ and ROC curves for these measurements and $r = \{1,3,5,7,9,15\}$ are shown in Fig.31.

	r=1	r=3	r=5	r=7	r=9	r=15
Mean	0.5910	0.6143	0.5875	0.6842	0.7273	0.7432
Variance	0.7736	0.9139	0.9648	0.9787	0.9761	0.9599
Skewness	0.8488	0.9320	0.9007	0.9180	0.9069	0.9266
Kurtosis	0.4032	0.7494	0.8155	0.8642	0.8967	0.9539
Minimum	0.6617	0.3937	0.3323	0.2928	0.2420	0.1730
Maximum	0.7453	0.9175	0.9800	0.9886	0.9871	0.9812
Entropy	0.8198	0.8525	0.9177	0.9436	0.9445	0.9129

Tab. 12: AUC for Laplacian transformation features for different values of the Laplacian window size parameter r

The AUC values of Laplacian transformation features indicate that these features are not capable of distinguishing between classes as well as some other evaluated features. We have shown that the performance of the features is strongly dependent on the size parameter h ($h = 2r + 1$) of the Laplacian convolution kernel. The results have shown that the response of these features is highest for the size of the convolution kernel $h = 15$, which corresponds to the approximate real size $R_A / 4$, where R_A is the estimated radius of a typical red blood cell in the input image. Since the Laplacian features reflect the intensity of regular particles fitting into the size of the convolution kernel, the results indicate that these features could be sensitive the young ring-form trophozoites, which are characterized by a distinct speckle of chromatin of approximately this size and that the AUC values might be higher if the positive class consisted only of the subset of these types of parasites. However, additional experiments would have to be performed to support this surmise.

9.4.8 Flat Texture Features

Flat texture transformation is calculated from the green channel of the extinction image with corrected shading. The same set of histogram measurements were calculated from the transformed image, including mean, variance, skewness, kurtosis, minimum and maximum represented by the seventh smallest and largest histogram values, and entropy. All these features were evaluated for a set of nine different sizes r of the square median operator window. Depending on the window size, the median transformation will smooth away all objects within the red blood cell up to the area of half the window area $r^2 / 2$. The flat texture then represents the difference between the original extinction image and the two-dimensional median filtered one. The measurements were calculated for the following set of sizes $r = \{3, 6, 9, 12, 15, 18, 25, 35, 45\}$.

The evaluation of flat texture features is implemented in the script `evalFlatTexture.m` and, similarly as for other evaluated features, all calculated feature values are saved in a mat-file with the same name together with vector named `targets` indicating the object's class, so that the result can be easily visualized at any time. The variable containing all the feature values is a three-dimensional array called `FT`. The dimensions of the array correspond to the individual samples, different values of the size parameter *r*, and histogram measurements, respectively.

The values of calculated AUCs are shown in Tab.13. Compared to the Laplacian transformation features, which were also evaluated for different values of the window size parameter, the dependence of the performance of the individual features on the size of the median operator window is much less pronounced for the flat texture features. This indicates that the size parameter is of lesser importance in this case. Moreover, lesser differences can be observed between the individual features compared to other image transformations.

In general, best performance can be observed for the variance followed by the maximum and the mean and for the sizes *r* of the median window between 15 and 25 pixels. Median filter with window size of 18 pixels, for which the highest value of AUC was observed, would smooth away all circular objects with diameter approximately up to $R_A / 4$, supposing there are no other objects in their proximity. It is interesting to see that this finding is consistent with the results observed during the evaluation of the Laplacian transformation features.

	r=3	*r*=6	*r*=9	*r*=12	*r*=15	*r*=18	*r*=25
Mean	0.9110	0.9750	0.9836	0.9839	0.9845	0.9900	0.9915
Variance	0.7649	0.9098	0.9390	0.9827	0.9919	0.9944	0.9917
Skewness	0.9145	0.9092	0.9186	0.9096	0.9025	0.9013	0.8928
Kurtosis	0.8545	0.8708	0.8778	0.8649	0.8578	0.8477	0.8461
Minimum	0.7729	0.9158	0.9487	0.9503	0.9497	0.9449	0.9451
Maximum	0.8542	0.9579	0.9680	0.9835	0.9883	0.9917	0.9900
Entropy	0.6883	0.8135	0.8608	0.9107	0.9400	0.9558	0.9740

Tab. 13: AUC for flat texture features for different values of the median operator window size r

The estimated probability density functions for all measurements are compared in Fig.32 for the median window size $r = 18$ and corresponding ROC curves for the same size of the median operator window are shown in Fig.33.

The flat texture features proved to provide comparably good ability to distinguish between

infected and non-infected red blood cells as the histogram features computed from the original extinction image or as the gradient transformation features. The best performance was achieved for the size of the median operator window corresponding approximately to the 10 / 3 of the typical non-infected red blood cell radius. Compared to the Laplacian transformation features, the results are less dependent on the exact value of the size parameter, which is in accordance with the principal differences of the underlying transformation methods.

9.4.9 Co-occurrence Features

Co-occurrence matrix is calculated from the green channel of the extinction image with the corrected non-uniform illumination, similarly as for most of the other texture features evaluated in this work. Furthermore, co-occurrence matrix is also computed from a flat texture image, which is obtained from the preprocessed extinction image. Parameters controlling the extraction of the co-occurrence matrix are the length of the displacement vector d and the number of gray-level values N_G determining the size of the co-occurrence matrix. Moreover, the normalization method determining how the intensity values in the input image are scaled to the specified number of gray levels N_G has to be defined. Commonly used methods include histogram equalization or linear spread function [29]. Intensity normalization is very important, because after reducing the number of intensity levels in the image, we could lose valuable texture information if all levels N_G are not used, for example, due to low contrast in the image. In our case, we can either treat each red blood cell sample individually regardless of the global intensity distribution in the input image or normalize the intensities for all red blood cells in the original extinction image. Alternatively, we could also normalize intensities for each image individually but with respect to the maximum and minimum intensity values in the input image, which is in case of the linear spread equal to the first method. Both approaches has shown to have certain drawbacks. The fundamental problem is that, especially in images containing red blood cells infected by malaria parasites in later stages of development filling the whole area of the cell, there is often large contrast between intensities of infected and non-infected red blood cells. If the intensities are scaled linearly according to the minimum and maximum intensity value in the image and, subsequently, the number of gray levels is reduced to N_G, we can lose important textural information in the infected red blood cell. This will happen if the textural information is concealed in limited range of intensities, which is not an exceptional case. On the other hand, if each sample is treated individually, linear spread of the intensities between the maximum and minimum intensity values in the sample image will result in extensive amplification of noise in non-infected red blood cells due to low contrast in the image. However, the textural information in the infected red blood cells will be preserved.

Histogram equalization can be used instead of the linear spread function to normalize intensities in the input images. This technique aims to create an image with equally distributed intensity levels over the whole intensity scale and enhances contrast for intensity values close to histogram maxima, and decreases contrast near minima. When histogram equalization is applied on the whole input image, contrast among the parasite pixels is reduced because these pixels correspond to histogram minima. Thus, the textural

information within the infected red blood cells is often smoothed away even without reducing the number of gray levels. If histogram equalization is applied on the individual red blood cell samples, contrast is extensively enhanced in non-infected red blood cells producing artificially-looking texture, which is typically more pronounced than in case of the linear spread.

All of the thirteen measurements given by Eq.53-65 were calculated for two different numbers of gray-levels $N_G = \{8, 16\}$, for ten different values of the distance parameter $d = \{1, 2, 3, 4, 5, 6, 8, 10, 12, 15\}$, and for six different images / normalization methods. The performance of the feature was evaluated using the following images.

- **IM1**: Extinction image with linearly spread intensity values between I_{min} and I_{max}, where I_{min} and I_{max} are the minimum and maximum intensity values in the particular red blood cell sample
- **IM2**: Extinction image with linearly spread intensity values between I_{Gmin} and I_{Gmax}, where I_{Gmin} and I_{Gmax} are the minimum and maximum intensity values in the whole original extinction image.
- **IM3**: Median filtered extinction image with size of the median operator window [3,3]. Intensities are automatically linearly scaled according to the minimum and maximum intensity values I_{min} and I_{max} in the particular red blood cell sample
- **IM4**: Extinction image with histogram equalization applied on individual red blood cell samples
- **IM5**: Extinction image with applied contrast-limited adaptive histogram equalization on the whole original extinction image.
- **IM6**: Flat texture image calculated using median operator window of size $r = 25$. Normalization is controlled automatically in each red blood cell sample according to the minimum and maximum intensity values in the image.

Median filtered image was included to evaluate the results after reducing the noise in the image. The fourth image is calculated as an alternative to the standard histogram equalization technique. Contrast-limited adaptive histogram equalization allows us to set a contrast enhancement limit to prevent over-saturation in homogeneous areas within non-infected red blood cells. Moreover, this method operates on small regions rather than the entire image and thus this method is a compromise between processing the whole image and the individual sample images. This technique is implemented by a Matlab function `adapthisteq` and the following parameters were used: `ClipLimit` = 0.05 and `NumTiles` = [20 20].

The co-occurrence matrix features are evaluated in the script `evalCooccurrence.m`. The values of the thirteen calculated features for all red blood cell sample images and for

all combinations of input parameters and images are stored in the mat-file `evalCooccurrence.mat` in a variable called `C`. This variable is a five-dimensional array whose dimensions correspond to the individual image samples, individual normalized images, distance vectors, calculated features f_1, …, f_{13}, and different sizes of the co-occurrence matrix, respectively.

The values of calculated AUCs indicate that the ability of these feature to distinguish between infected and non-infected red blood cells is smaller compared to most of the other evaluated features. The highest achieved value of AUC was 0.9831. The reason for lower performance could be seen in the fact that the co-occurrence matrix describes distribution of co-occurring values at a given offset and, therefore, is better suited for description of highly regular textures. Although some infected red blood cells exhibit relatively regular texture properties, such infected cells are represented only by a relatively small number of samples in our dataset. Major part of the infected red blood cell dataset comprises of parasites in early stages of development, which exhibit little regularity in texture. Such infected red cells might also be hard to distinguish from the non-infected cells due to the amplified noise in the non-infected red blood cells. If the dataset contained more samples, more experiments could be performed to show whether the co-occurrence features could possibly distinguish between different types of parasites.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	max
IM1	0.8104 $d=1$	0.9468 $d=1$	0.8504 $d=8$	0.9527 $d=1$	0.9400 $d=1$	0.9527 $d=1$	0.9446 $d=1$	0.7792 $d=1$	0.8103 $d=1$	0.9473 $d=1$	0.9459 $d=1$	0.8472 $d=1$	0.6858 $d=15$	**0.9527**
IM2	0.7257 $d=15$	0.8949 $d=10$	0.8539 $d=15$	0.8528 $d=15$	0.7912 $d=10$	0.8528 $d=15$	0.8813 $d=3$	0.7824 $d=15$	0.7910 $d=15$	0.8995 $d=10$	0.8347 $d=15$	0.3078 $d=1$	0.8256 $d=2$	**0.8995**
IM3	0.7752 $d=2$	0.8994 $d=1$	0.8645 $d=12$	0.9643 $d=1$	0.8616 $d=1$	0.9643 $d=1$	0.9587 $d=1$	0.7601 $d=1$	0.7717 $d=1$	0.9019 $d=1$	0.8808 $d=1$	0.7614 $d=1$	0.7380 $d=12$	**0.9643**
IM4	0.7653 $d=15$	0.8302 $d=15$	0.7876 $d=1$	0.8240 $d=15$	0.7133 $d=15$	0.8240 $d=15$	0.8238 $d=15$	0.7601 $d=3$	0.7851 $d=12$	0.8308 $d=15$	0.8349 $d=15$	0.7137 $d=1$	0.7244 $d=1$	**0.8349**
IM5	0.7952 $d=12$	0.9167 $d=12$	0.8393 $d=2$	0.8162 $d=15$	0.8574 $d=12$	0.8162 $d=15$	0.8288 $d=15$	0.8343 $d=6$	0.8460 $d=10$	0.9188 $d=12$	0.8902 $d=12$	0.1978 $d=1$	0.8308 $d=1$	**0.9188**
IM6	0.3582 $d=1$	0.4269 $d=2$	0.3066 $d=15$	0.3362 $d=1$	0.4336 $d=2$	0.3362 $d=1$	0.3268 $d=1$	0.7960 $d=1$	0.8449 $d=1$	0.4261 $d=2$	0.9264 $d=3$	0.4721 $d=4$	0.9664 $d=4$	**0.9664**
max	**0.8104**	**0.9468**	**0.8645**	**0.9643**	**0.9400**	**0.9643**	**0.9587**	**0.8343**	**0.8460**	**0.9473**	**0.9459**	**0.8472**	**0.9664**	***0.9664***

Tab. 14: Maximum AUC values for co-occurrence matrix of size $N_G = 8$ obtained for individual images IM1, …, IM6 and features f_1, …, f_{13}. Corresponding lengths d_m of the displacement vector, for which this maximum was observed, are shown below the AUC value.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	max
IM1	0.8640 $d=1$	0.9531 $d=1$	0.8569 $d=15$	0.9531 $d=1$	0.9444 $d=1$	0.9531 $d=1$	0.9444 $d=1$	0.7858 $d=1$	0.8528 $d=1$	0.9531 $d=1$	0.9491 $d=1$	0.8623 $d=1$	0.6621 $d=15$	**0.9531**
IM2	0.8091 $d=15$	0.9828 $d=8$	0.8537 $d=1$	0.8155 $d=15$	0.8831 $d=12$	0.8155 $d=15$	0.8389 $d=1$	0.8805 $d=15$	0.8881 $d=15$	0.9831 $d=8$	0.9478 $d=12$	0.7733 $d=1$	0.8924 $d=1$	**0.9831**
IM3	0.8124 $d=2$	0.9218 $d=1$	0.8672 $d=15$	0.9648 $d=1$	0.8611 $d=1$	0.9648 $d=1$	0.9589 $d=1$	0.7809 $d=1$	0.7977 $d=1$	0.9226 $d=1$	0.8805 $d=1$	0.8052 $d=1$	0.7323 $d=12$	**0.9648**
IM4	0.7866 $d=2$	0.8392 $d=15$	0.6447 $d=1$	0.8261 $d=15$	0.7411 $d=2$	0.8261 $d=15$	0.8272 $d=1$	0.6706 $d=3$	0.7773 $d=1$	0.8392 $d=15$	0.8308 $d=15$	0.7295 $d=1$	0.7125 $d=1$	**0.8392**
IM5	0.8308 $d=15$	0.9400 $d=12$	0.8275 $d=2$	0.7851 $d=15$	0.8611 $d=12$	0.7851 $d=15$	0.7966 $d=15$	0.8691 $d=15$	0.8790 $d=15$	0.9403 $d=12$	0.9084 $d=15$	0.1050 $d=11$	0.8861 $d=1$	**0.9403**
IM6	0.3881 $d=1$	0.4254 $d=2$	0.3197 $d=15$	0.3360 $d=1$	0.4395 $d=1$	0.3360 $d=1$	0.3243 $d=1$	0.7687 $d=1$	0.8765 $d=1$	0.4251 $d=2$	0.9278 $d=2$	0.4785 $d=4$	0.9723 $d=4$	**0.9723**
max	**0.8640**	**0.9828**	**0.8672**	**0.9648**	**0.9444**	**0.9648**	**0.9589**	**0.8805**	**0.8881**	**0.9831**	**0.9491**	**0.8623**	**0.9723**	***0.9831***

Tab. 15: Maximum AUC values for co-occurrence matrix of size $N_G = 16$ obtained for individual images IM1, ..., IM6 and features f_1, ..., f_{13}. Corresponding lengths d_m of the displacement vector, for which this maximum was observed, are shown below the AUC value.

In summary, better results were generally achieved for features calculated from co-occurrence matrix with $N_G = 16$. The maximum AUC calculated for $N_G = 8$ was $\max(\mathrm{AUC}_{N_G=8}) = 0.9664$ compared to $\max(\mathrm{AUC}_{N_G=16}) = 0.9831$. Although the results for the two numbers of gray levels were usually not significantly different, for example in the histogram equalized image IM4, noticeably better results were observed in feature f_3 for $N_G = 8$. Rather inconsistent results were observed in the image IM2 where some features showed significantly better result for $N_G = 16$ while the others showed better results for $N_G = 8$ for all values of the distance parameter d.

Maximum AUC values for individual images IM1, …, IM6 and features f_1, …, f_{13}, and the corresponding lengths d_m of the displacement vector, for which this maximum was observed, are displayed in Tab.14 for $N_G = 8$ and in Tab.15 for $N_G = 16$. Dependency of the feature performance on the displacement parameter d was usually consistent and the values of AUC are gradually decreasing for $d > d_m$ and $d < d_m$. The results shown in Tab.14 and 15 confirm predicted strong dependence on the combination of the number of gray levels and the method of scaling the gray-scale intensities into the reduced number of gray levels N_G. This makes these features even less useful because if a classification method is devised based on these features, the performance of the classifier may deteriorate with the change of some characteristics, such as contrast or noise, in the input image data set. In general, the worst results were observed for the histogram equalized image IM4, which is consistent with our prediction. However, the contrast-limited histogram equalization technique (IM5) also did not prove superior to the other normalization methods. Judging by the maximum values of AUC, flat texture image also delivered good results compared to other images used. This, however, holds true only for few of the measurements. By contrast, most of the other features calculated from the flat texture image produced results with extremely small AUCs. Although the highest values of AUCs were observed for the image IM2 when sixteen gray levels were used, the results were not at all that good when eight gray levels

were used. The reason for this behavior may be seen in the limited range of intensities in which the textural details are concealed. When small number of gray levels is used, these details may be smoothed away as discussed earlier in this section.

9.4.10 Run-length features

Run-length features given by Eq.71-81 are calculated for four different numbers of gray levels $N_G = \{8, 16, 32, 64\}$ and for the same set of images IM1-IM6 with normalized intensities that were used for calculation of co-occurrence matrices. The extinction image is obtained from the green channel of the original transmission image with correction of non-uniform illumination. The same problems regarding the intensity normalization we encountered in the previous section also apply in the case of run-length features.

Run-length features are evaluated in the script `evalRunLength.m`. A four-dimensional array `RL` containing the results of the calculation is stored in a mat-file `evalRunLength.mat`. The dimensions of this array correspond to the individual red blood cell samples, six images with normalized intensities, eleven run-length features, and four numbers of different gray levels, respectively.

The maximum AUC values for individual images IM1-IM6 and numbers of gray levels N_G are shown in Tab.16. In general, the best results were obtained for the first three images IM1-IM3. The performance of the run-length features derived from the images with equalized histogram is quite low and the AUCs for the features derived from the flat texture image are extremely small. Regarding the number of gray levels, the best results depend on the particular measurement and normalization method. In general, for images IM1 and IM3, the differences between the AUCs for the individual numbers of gray levels were relatively low for many measurements, while for IM2 the results were generally better for higher numbers of gray levels. It also has to be remarked that the highest AUC obtained for the image IM2 and $N_G = 64$ was rather an exception. For the overwhelming majority of the calculated features, the AUC was not greater than 0.98. This again indicates that the performance of the run-length features, similarly as for the co-occurrence matrix features, is strongly dependent on the combination of the normalization method and number of gray levels used. This makes these features rather problematic to use due to the low generalization properties as noted in the previous section.

Comparing the individual measurements, better results were generally obtained for features LGRE (f_6), HGRE (f_7), SRLGE (f_8), SRHGE (f_9), and LRLGE (f_{10}), although the results are again dependent on the particular image, from which the run-length matrix is generated, and the number of gray levels used.

	$N_G = 8$	$N_G = 16$	$N_G = 32$	$N_G = 64$
IM1	0.9759 (f_6)	0.9762 (f_8)	0.9758 (f_6)	0.9761 (f_6)
IM2	0.9449 (f_7)	0.9775 (f_7)	0.9870 (f_9)	0.9921 (f_9)
IM3	0.9796 (f_6)	0.9797 (f_6)	0.9797 (f_6)	0.9799 (f_8)
IM4	0.7774 (f_7)	0.8323 (f_9)	0.7923 (f_9)	0.6921 (f_{11})
IM5	0.8312 (f_7)	0.8535 (f_9)	0.8331 (f_7)	0.8326 (f_7)
IM6	0.4708 (f_5)	0.4516 (f_5)	0.4092 (f_5)	0.4067 (f_5)

Tab. 16: Maximum AUC values of run-length features for individual images IM1-IM6 and numbers of gray levels N_G. Corresponding measurement, for which the maximum was observed, is shown in the parentheses.

10 Conclusions

In this work, we present a method for segmentation of red blood cells in microscopic blood smear images and describe a designed graphical user interface for using the segmentation method. The main purpose of the GUI is to execute the segmentation method, manually correct its results, label the samples and save the information about individual segmented cells. Using this GUI, we have created a database of labeled red blood cells containing non-infected cells and cells infected by malaria parasites. A set of features is proposed to be extracted from the area of a red blood cell in order to detect malaria. The performance of these features is evaluated on the created red blood cell database using ROC curves.

The segmentation method presents a relatively simple solution of separating red blood cells from the background and isolating overlapping or occluded cell. The method is based on processing of a binary image obtained by thresholding and utilizes the watershed transformation for separating cell compounds. This approach was chosen as more elaborate techniques tested in our previous work did not perform optimally due to relatively high variations in qualitative characteristics of the input images. The method produced good results on all input images with only occasional over-segmented cells, which were easily merged in the GUI tool.

The designed GUI provides all tools necessary for correcting the contours of segmented cells, labeling of samples, manipulating with files, and saving the results. During our work, it proved to be a very useful and practical tool. It has been designed with respect to easy extendability. It enables a user to register new segmentation methods and new labels and as it is developed in Matlab, virtually any part of the GUI can easily be modified to conform to the user's requirements. For frequent work, certain improvements could be made in the GUI, for example implementation of some semi-automatic methods for the correction of cell contours, a set of hot keys, the use of mouse wheel for zooming, etc. However, some of these improvements are also limited by the programing environment.

The created database of red blood cells is implemented in an easily accessible way so that it could be readily used in other projects. It contains 1694 non-infected red blood cell samples and 117 samples infected by different species of malaria parasites in various stages of development. This is an acceptable number to evaluate performance of the proposed features on the problem of distinguishing between infected and non-infected cells but not for separating more classes, for example different species of Plasmodium parasites. In this case, we were limited by a relatively small number of available blood smear images. The designed GUI, however, allows one to very easily add new samples.

There are many possible classification scenarios in the problem of malaria diagnosis. In this work, we focused mainly on general evaluation of the features and we tested their discrimination power on the problem of distinguishing between infected and non-infected red blood cells. The detection of infected cells is the primary task and usually has to be carried out before any further analysis can be performed. In our case, any further analysis of the infected samples was also limited by the relatively small number of infected red blood cells samples. The results have shown that very good discrimination between the two

classes can be obtained using histogram features, especially the maximum (minimum in case of the transmission image) represented, in our case, by the seventh largest (smallest) histogram value, gradient transformation features, and flat texture features. The features based only on the shape of the cell did not prove to be very useful for the general problem of distinguishing between infected and non-infected cells. However, we have shown, that they could possibly be used in some special cases, for example to detect mature Plasmodium falciparum gametocytes, which have characteristic crescent-like shape. The co-occurrence matrix features and run-length features have provided rather inconsistent results. Although they are clearly capable to distinguish the classes to some extent, the results strongly depend on the intensity normalization method and the number of gray levels used. The intensity normalization itself proved to provide relatively inconsistent results, mainly for non-infected red blood cells, and might constitute the major problem. For the observed issues, using these features without combining them with some other features is not suggested as long as a more reliable normalization method for this particular case is proposed. They could possibly be used for further analysis of infected cells as these cells have more distinct texture and, in this case, intensity normalization provided more consistent results. This would, however, require further tests using larger database of infected red blood cell samples.

References

[1] Springl, V.: *Automatic Malaria Diagnosis through Microscopy Imaging – semestral project*.

[2] Di Ruberto, C., et al: *Morphological Image Processing for Evaluating Malaria Disease*. IWVF4. 2001, 739-748.

[3] Di Ruberto, C., et al: *Analysis of infected blood cell images using morphological operators*. Image and Vision Computing 20(2). 2002, 133-146

[4] Ross, N.E., Pritchard, C.J., Rubin, D.M., Dusé, A.G.: *Automated image processing method for the diagnosis and classification of malaria on thin blood smears*. Medical and Biological Engineering and Computing 44, 2006, 427-436

[5] Rodenacker, K., Bengtsson, E.: *A feature set for cytometry on digitized microscopic images*. Analytical Cellular Pathology 25, 2003, 1-36

[6] Pinzón R., Garavito G., Hata Y., Arteaga L., García J.D., *Development of an automatic counting system for blood smears*. Congress of the Spanish Biomedical Engineering Society (CASEIB 2004), Santiago de Compostela, Nov 2004, pp 45 - 49.

[7] Sonka, M., Hlavac, M., Boyle, R.: *Image Processing, Analysis, and Machine Vision*. Brooks / Cole. 1998

[8] Otsu, N., *A Threshold Selection Method from Gray-Level Histograms*. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[9] Canny, J.: *A Computational Approach to Edge Detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, 1986, pp. 679-698.

[10] Soille, P.: *Morphological Image Analysis: Principles and Applications.* Springer-Verlag, Heidelberg, 2003.

[11] Sedgewick, R.: *Algorithms in C*, 3rd Ed., Addison-Wesley, 1998, pp. 11-20.

[12] Jain R., Kasturi R., Schunck B.: *Machine Vision*, McGraw-Hill, 1995

[13] Wu Q., Merchant F., Castleman K.: *Microscope Image Processing*, Academic Press, 2008

[14] Tek, F.B., Dempster, A.G., Kale, I.: *Blood Cell Segmentation Using Minimum Area Watershed And Circle Radon Transformations*, Mathematical Morphology: 40 Years On, 441-454, Springer, 2005.

[15] Chen, Q., Yang, X., Petriu, E.M.: *Watershed Segmentation for Binary Images with Different Distance Transforms*, IEEE, 2004, pp. 111-116

[16] Pizer, S.M. et al.: *Adaptive Histogram Equalization and its Variations*. Computer Vision, Graphics and Image Processing, vol. 39, 1987, pp. 355–368.

[17] Díaz, G., Gonzalez, F., Romero, E., *Infected Cell Identification in thin Blood Images Based on Color Pixel Classification: Comparison and Analysis*. Springer Berlin, 2007, pp. 812-821

[18] Theodoridis, S., Koutroumbas K., *Pattern Recognition*. Academic Press. 2006

[19] Hu, M. K.: *Visual Pattern Recognition by Moment Invariants*, IRE Trans. Info. Theory, vol. IT-8, pp.179-187, 1962.

[20] Reiss, T.H.: *Recognizing Planar Objects Using Invariant Image Features*. Springer-Verlag, Berlin, 1993.

[21] Tek, F., Dempster, A., Kale, I.: *Malaria parasite detection in peripheral blood images*. Proceeding of British Machine Vision Conference. 2006

[22] Tek, F., Dempster, A, Kale, I.: *A colour normalization method for giemsa-stained blood cell images.* In Proc. Sinyal Isleme ve Iletisim Uygulamalari, April 2006.

[23] Pitas, I.: *Digital Image Processing Algorithms and Applications*. Wiley-IEEE, 2000

[24] Haralick, R.M., Shanmugam, K., Dinstein, I.: *Textural features for image classification. IEEE Trans. on Systems*, Man, and Cybernetics SMC-3 (1973), 610–621.

[25] Xu, D.H., Kurani, A., Furst, J.D. , Raicu, D.S.: *Run-length encoding for volumetric texture*. The Fourth IASTED International Conference on Visualization, Imaging, and Image Processing-VIIP 2004, Marbella, Spain, 2004, pp. 452-458.

[26] Tang, X.: *Texture Information in Run-Length Matrices*. IEEE Transactions on Image Processing, vol. 7, no. 11, November 1998, pp. 1602-1609

[27] Seong, H.P., Jin, M.G., Chan-Hee, J.: *Receiver Operating Characteristic (ROC) Curve: Practical Review for Radiologists*. Korean J Radiol. 2004, pp. 11-18

[28] Bradley, A.P.: *The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms*. Pattern Recognition, Vol. 30, Issue 7, July 1997, pp. 1145-1159

[29] Rodenacker, K.: *Invariance of textural features in image cytometry under variation of size and pixel magnitude*. Anal.Cell. Pathol. 8 (1995), 117–133.

[30] WMR UNICEF, *World Malaria Report.* Technical Report, WMR and UNICEF, 2005.

[31] Makler M.T., Palmer C.J., Alger A.L.: *A review of practical techniques for the diagnosis of malaria.* Annals of Tropical Medicine and Parasitology 92(4), 1998, 419-433

[32] Hisaeda, H., Yasutomo, K., Himeno, K. *Malaria: immune evasion by parasite*. Int. J. Biochem. Cell Biol. 37, 700-706.

[33] Smyth, J. D., *Introduction to Animal Parasitology.* Cambridge University Press, Cambridge, 1994.

[34] Mui, J.K., Fu, K.S.: *Automated classification of nucleated blood cells using a binary tree classifier.* IEEE Trans. Pattern Analysis and Machine Intelligence 2(5), 1980, 429-443

[35] *Centers for Disease Control and Prevention: Public Health Image Library* [online]. 2005 WWW: <http://phil.cdc.gov/phil/home.asp>.

[36] Warhurst, D.C., Williams, J.E.: *Laboratory diagnosis of malaria*. J Clin Pathol 49, 1996, 533–38

[37] Angulo, J., Flandrin, G.: *Automated Detection of Working Area of Peripheral Blood Smears Using Mathematical Morphology.* Analytical Cellular Pathology. 2003, s. 37-49

[38] Ree, G.H., Sargeaunt, P.G.: *Laboratory diagnosis of malaria*. BMJ 1976, 1-152

[39] Pammenter, M.D.: *Techniques for the diagnosis of malaria*. S Afr Med J 74(2), 1988, pp. 55-57

[40] Oaks, S.C.J. et al: *Malaria: obstacles and opportunities.* National Academy Press, Washington, DC. 1991

[41] Bloland, P.B.: *Drug resistance in malaria.* WHO/CDS/CSR/DRS/ 2001.4. World Health Organization, Switzerland, 2001

[42] Mitiku, K., Mengistu, G., Gelaw, B.: *The reliability of blood film examination for malaria at the peripheral health unit*. Ethiop.J.Health Dev, 17(3), 2003, pp. 197 – 204.

[43] Gonzalez, R.C., Woods, R.E., Eddins, S.L., *Digital Image Processing Using MATLAB*. Prentice Hall, 2004

[44] Marchand, P., Holland, T. O.: *Graphics and GUIs with MATLAB,* 3rd ed. Chapman & Hall/CRC, 2003.

[45] Edgar, G.A.: *Measure, Topology and Fractal Geometry.* Springer-Verlag, 1995

[46] Sio, W.S.S, et al: *MalariaCount: An image analysis-based program for the accurate determination of parasitemia.* Journal of Microbiological Methods. 2006

Appendices

Appendix A – Probability Density Functions and ROC Curves

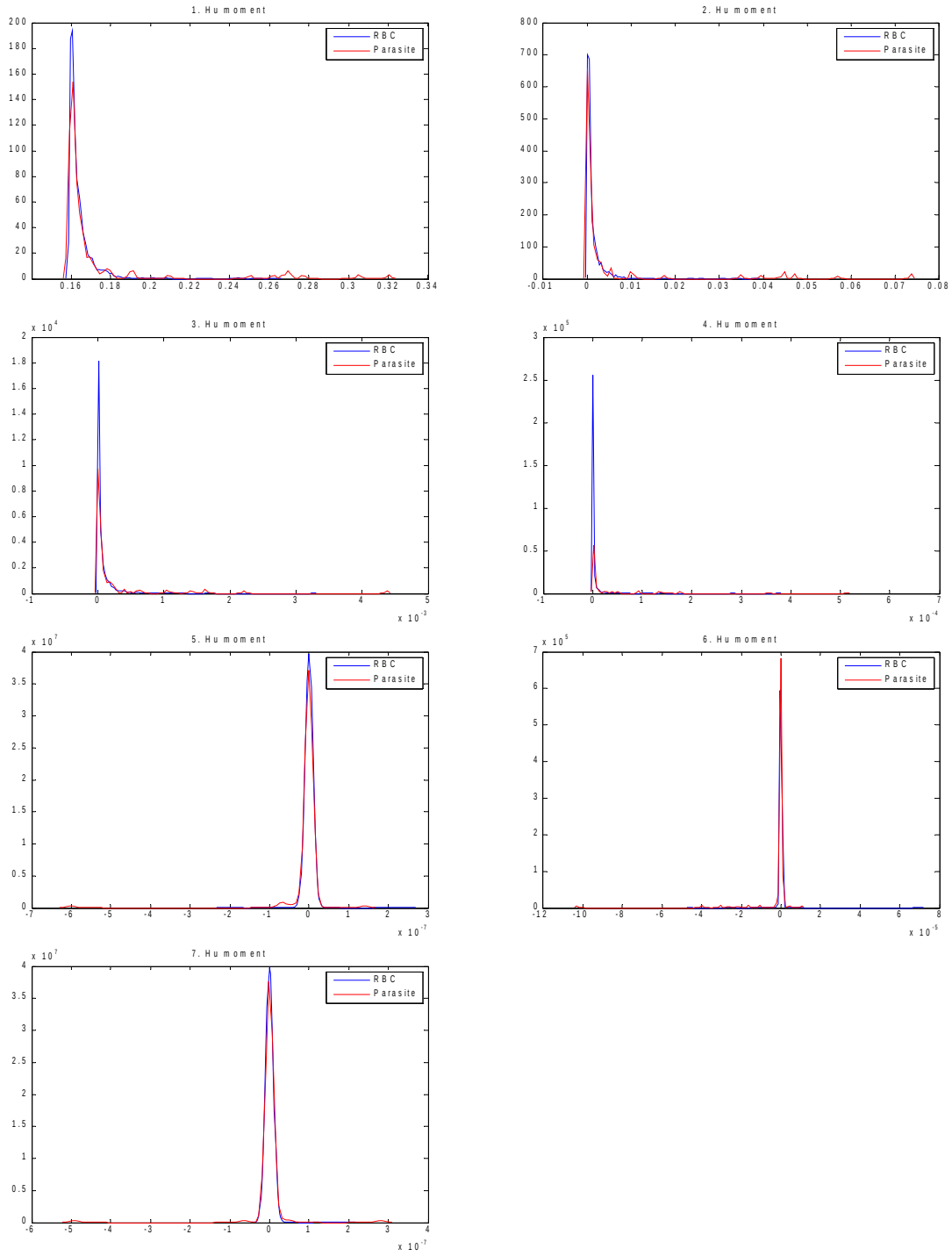


Fig. 17: Estimated probability density functions for Hu set of invariant features

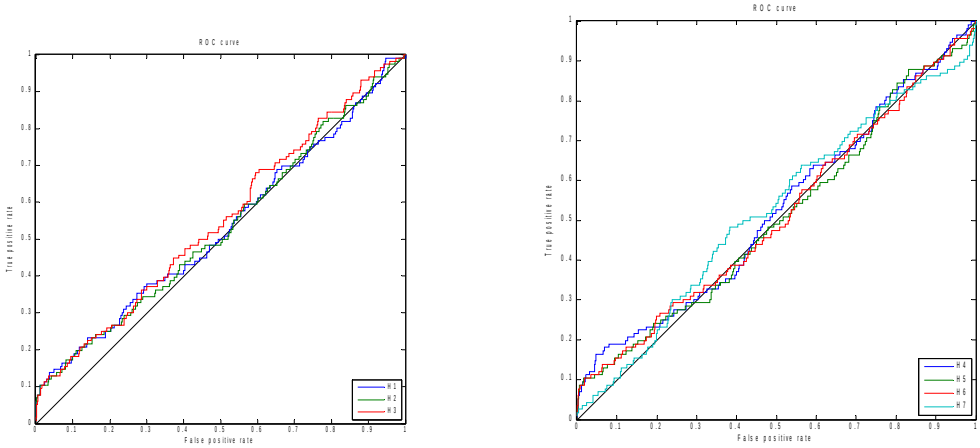


Fig. 18: ROC curves for Hu set of invariant moment features

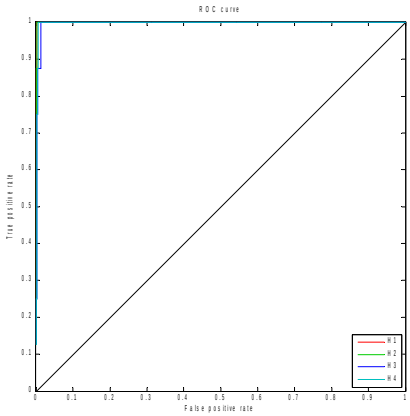


Fig. 19: ROC curves for Hu set of invariant moment features H1 – H4 calculated using non-infected red blood cells and red blood cells infected by mature Plasmodium Falciparum parasites

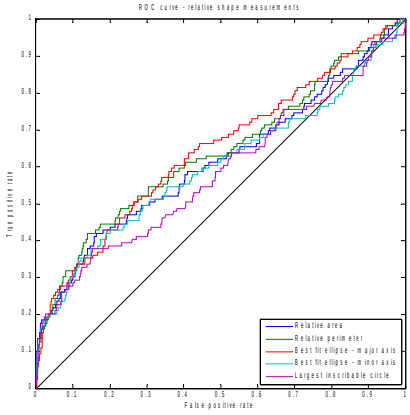


Fig. 20: ROC curves for the relative shape measurement features

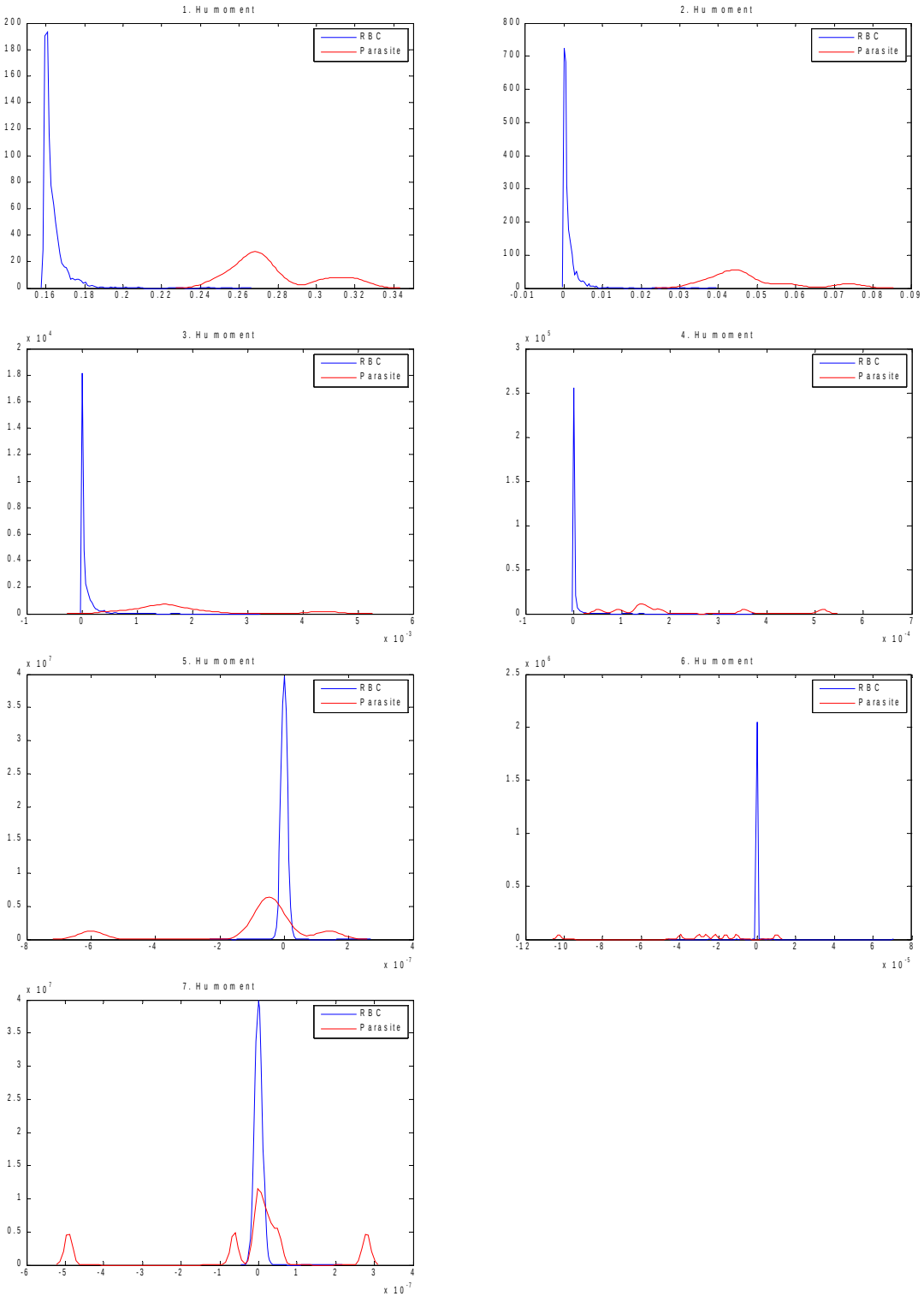


Fig. 21: Estimated probability density functions for Hu set of invariant features for non-infected red blood cells and red blood cells infected by mature Plasmodium falciparum parasites

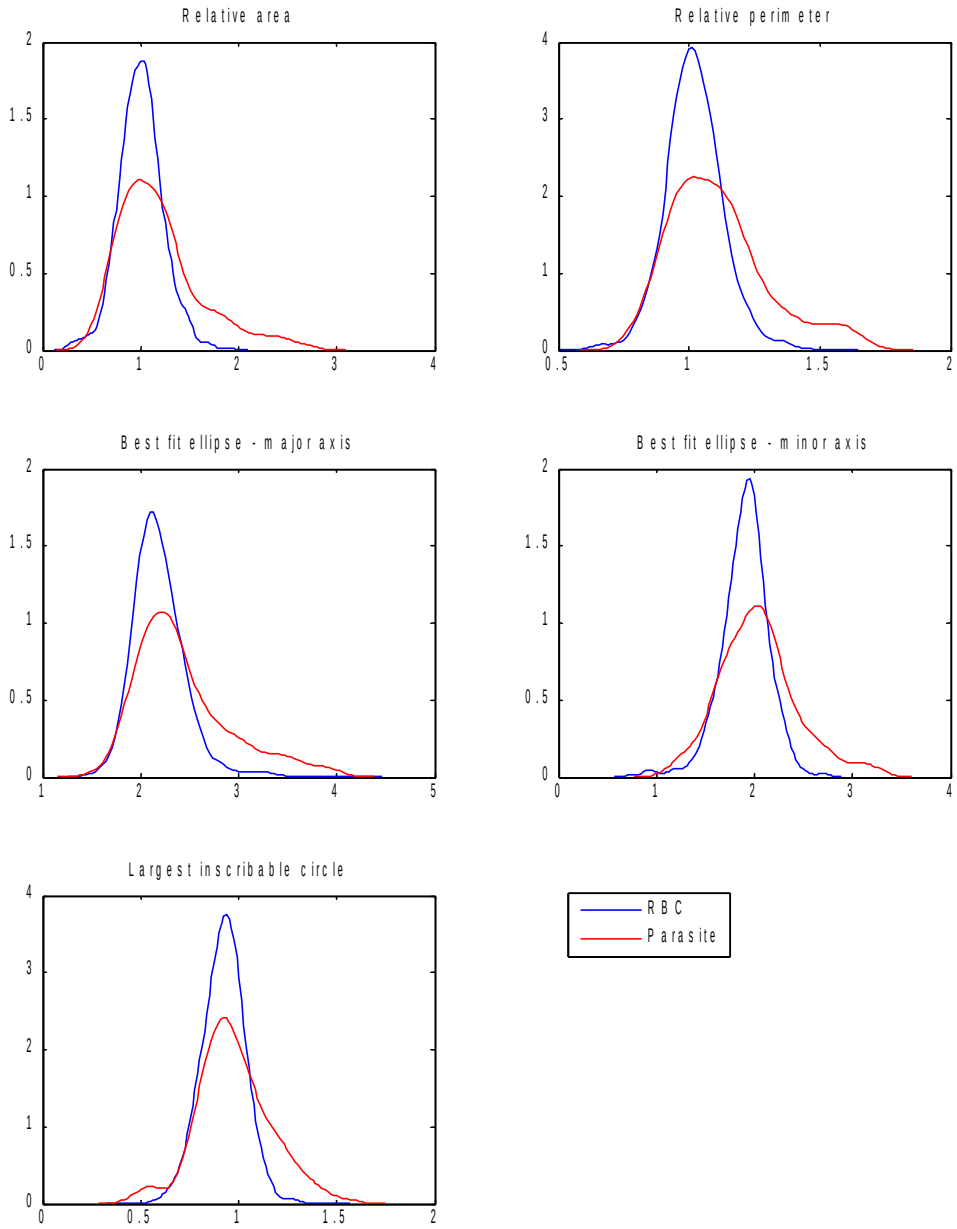


Fig. 22: Estimated probability density functions for the relative shape measurement features

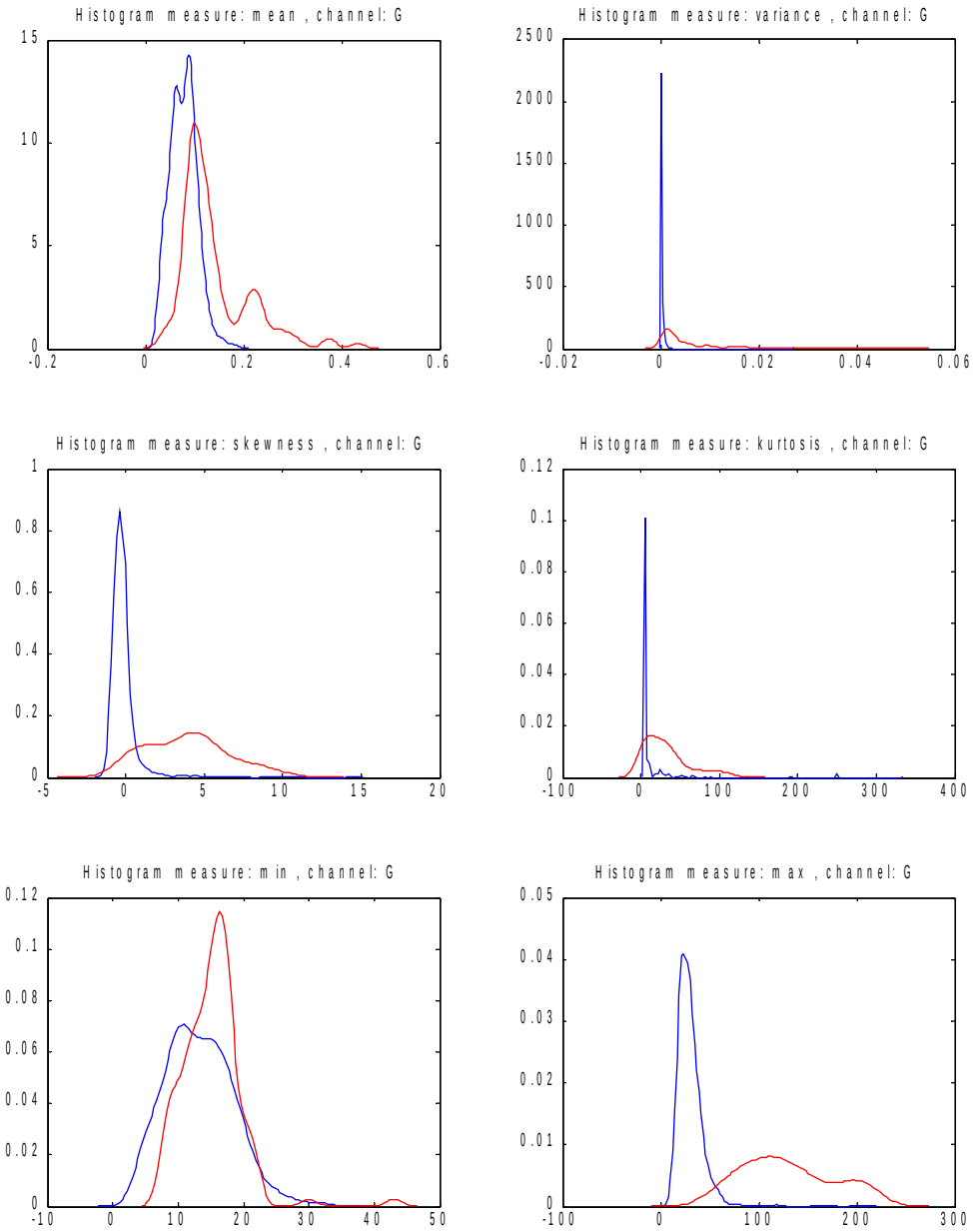


Fig. 23: Estimated probability density functions for histogram-based features calculated from the green channel of the extinction image (mean, variance, skewness, kurtosis, minimum, maximum)

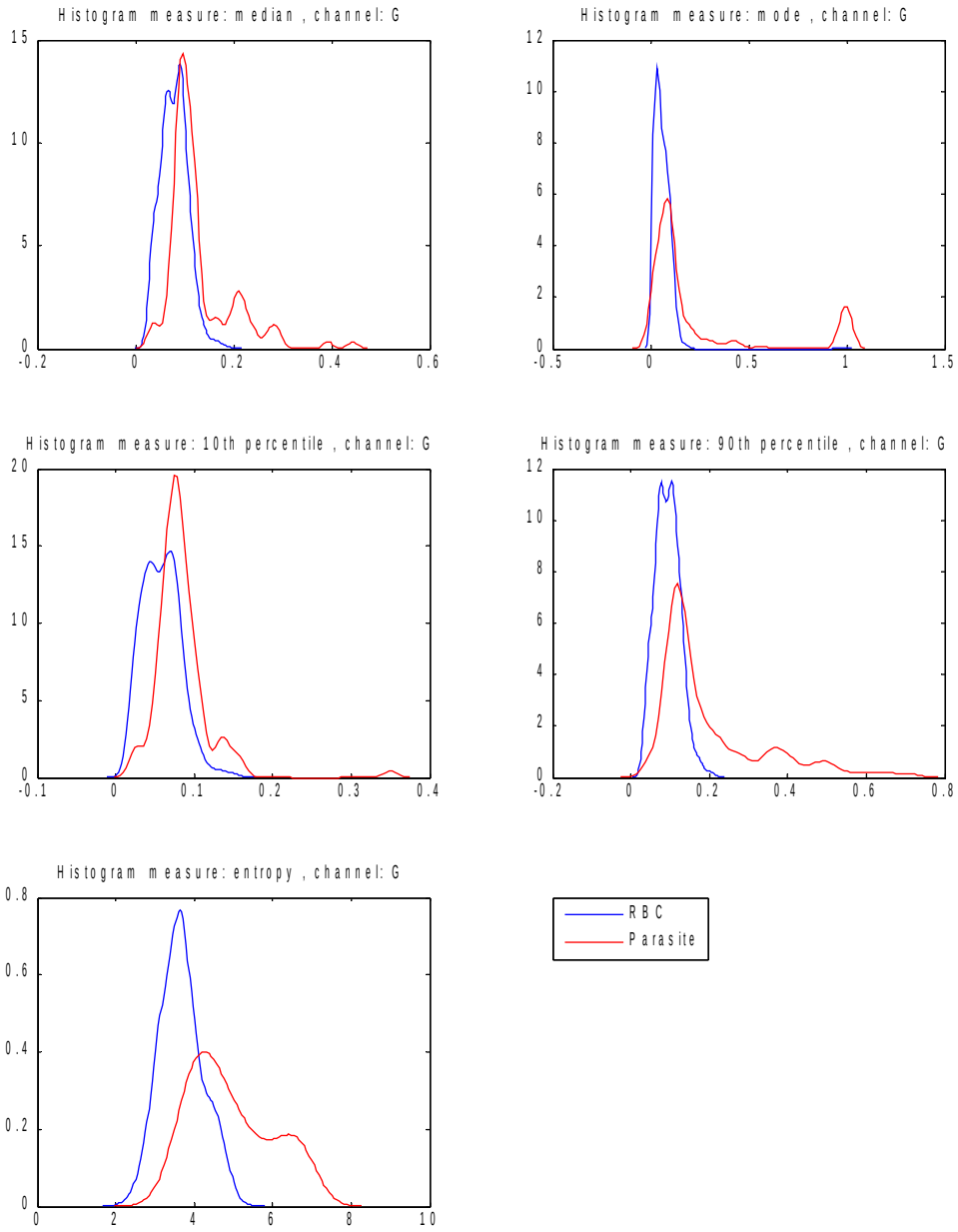


Fig. 24: Estimated probability density functions for histogram-based features calculated from the green channel of the extinction image (median, mode, 10th percentile, 90th percentile, entropy)

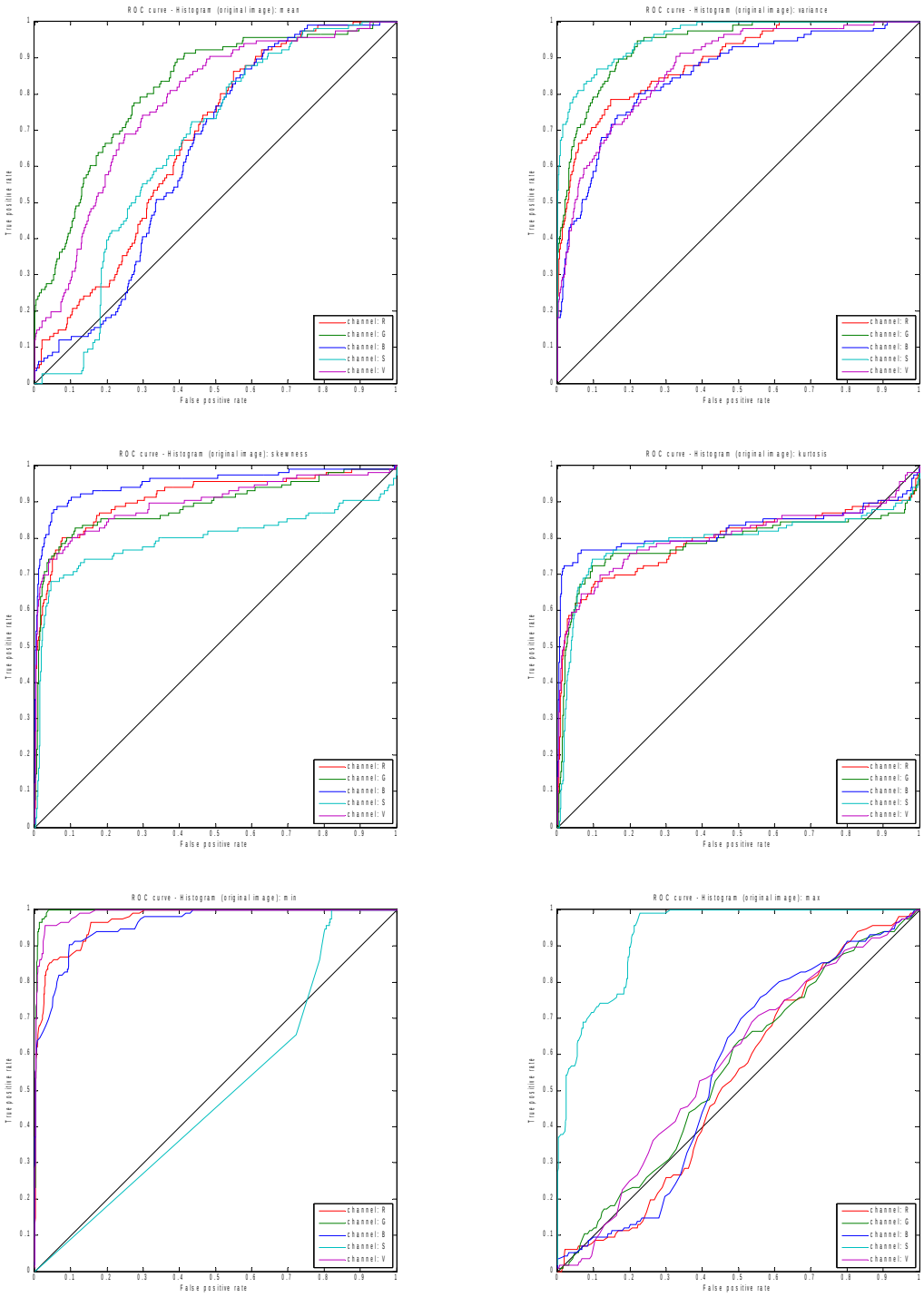


Fig. 25: ROC curves for the histogram-based features calculated from the original transmission image (mean, variance, skewness, kurtosis, minimum, maximum)

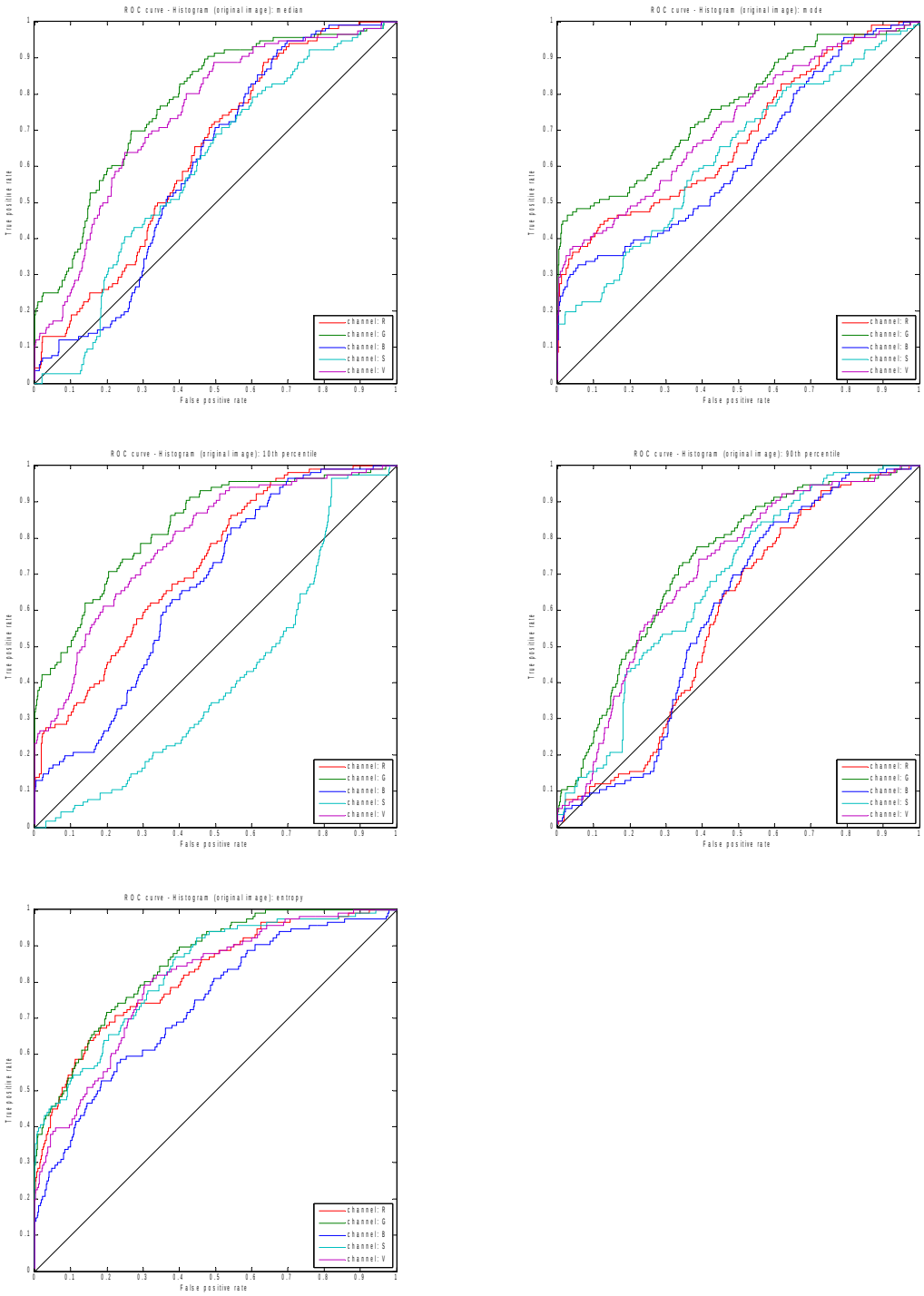


Fig. 26: ROC curves for the histogram-based features calculated from the original transmission image (median, mode, 10th percentile, 90th percentile, entropy)

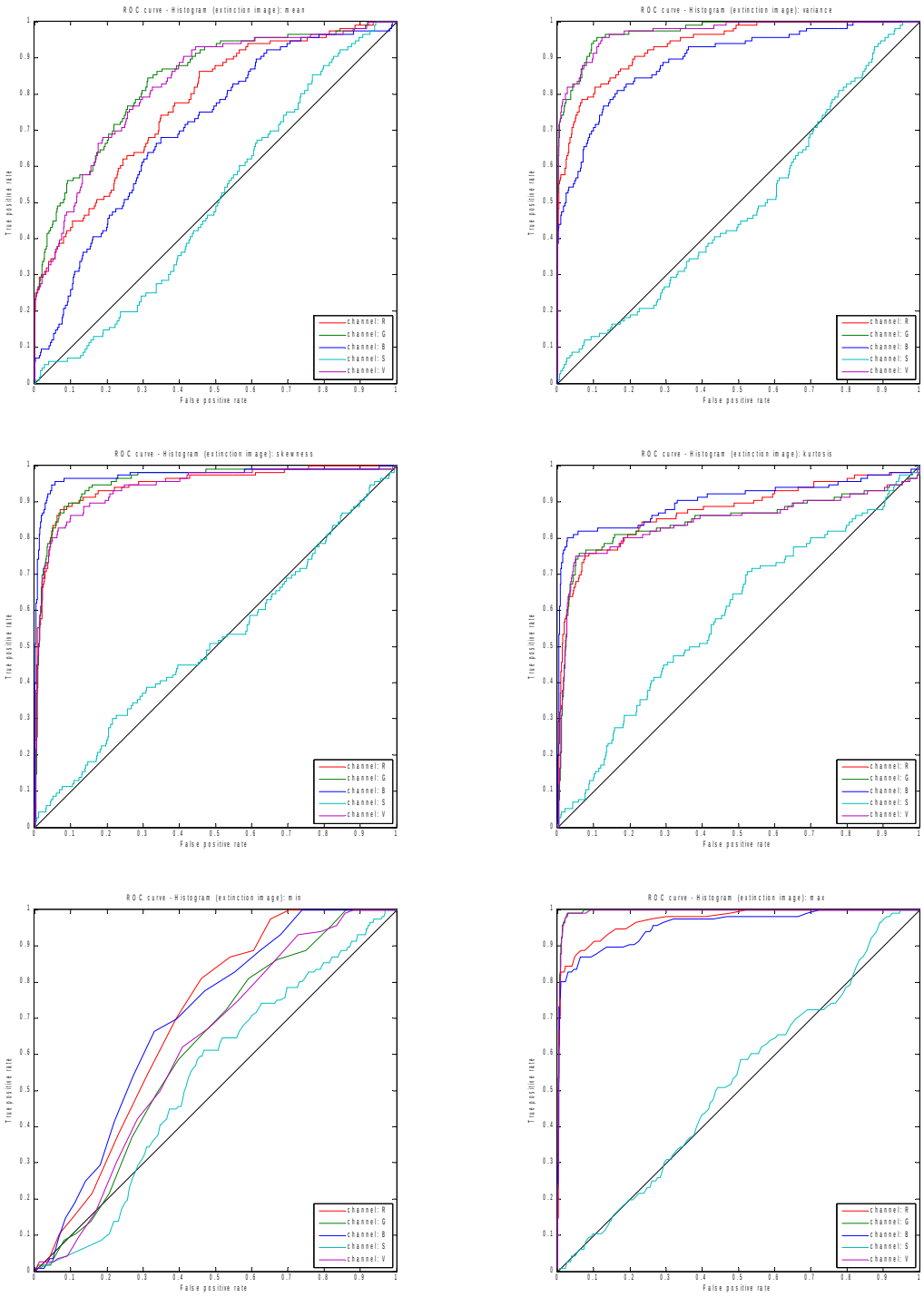


Fig. 27: ROC curves for the histogram-based features calculated from the extinction image (mean, variance, skewness, kurtosis, minimum, maximum)

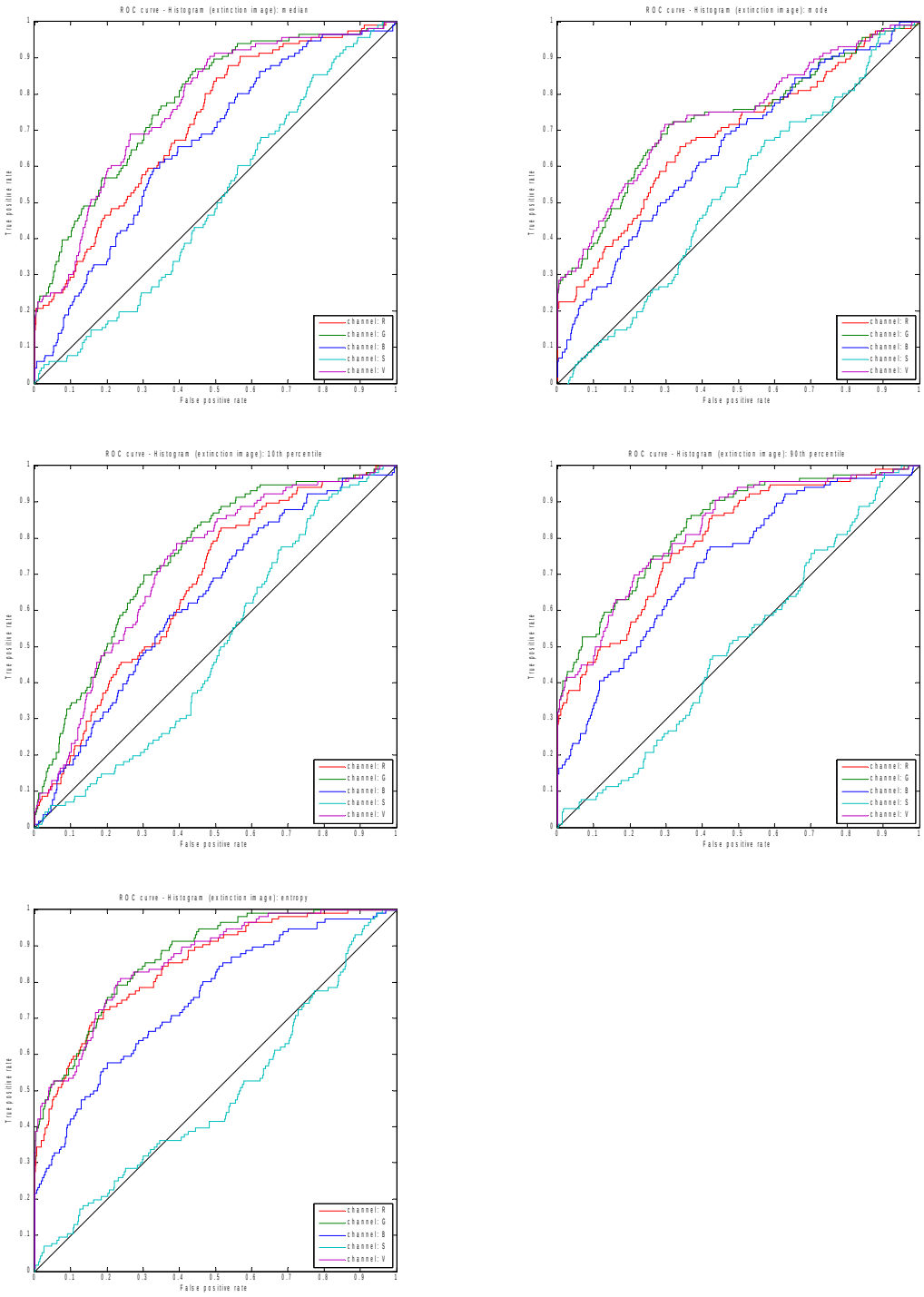


Fig. 28: ROC curves for the histogram-based features calculated from the extinction image (median, mode, 10th percentile, 90th percentile, entropy)

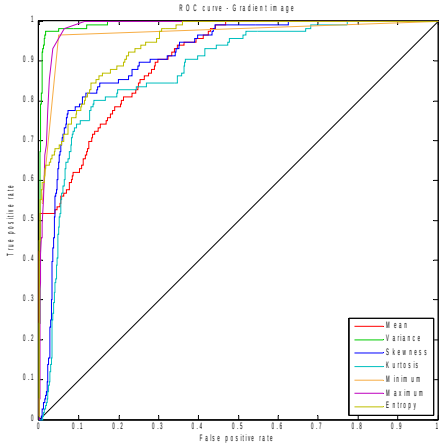


Fig. 29: ROC curves for gradient transformation features

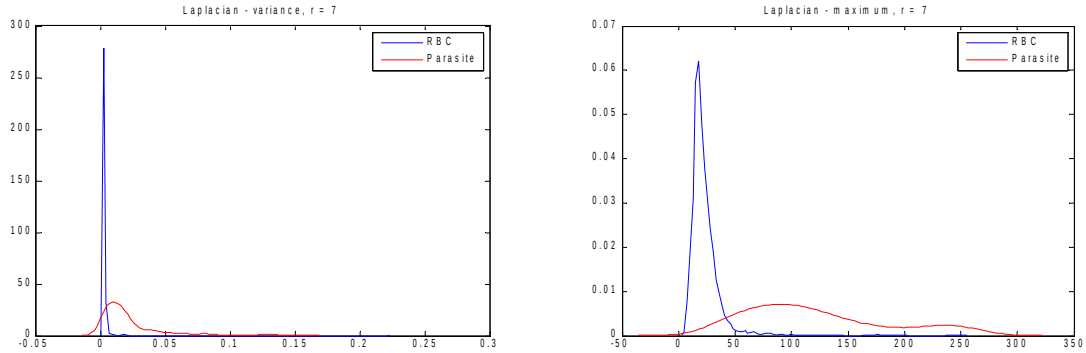


Fig. 30: Estimated probability density functions for Laplacian transformation features and measurements variance and maximum for r = 7

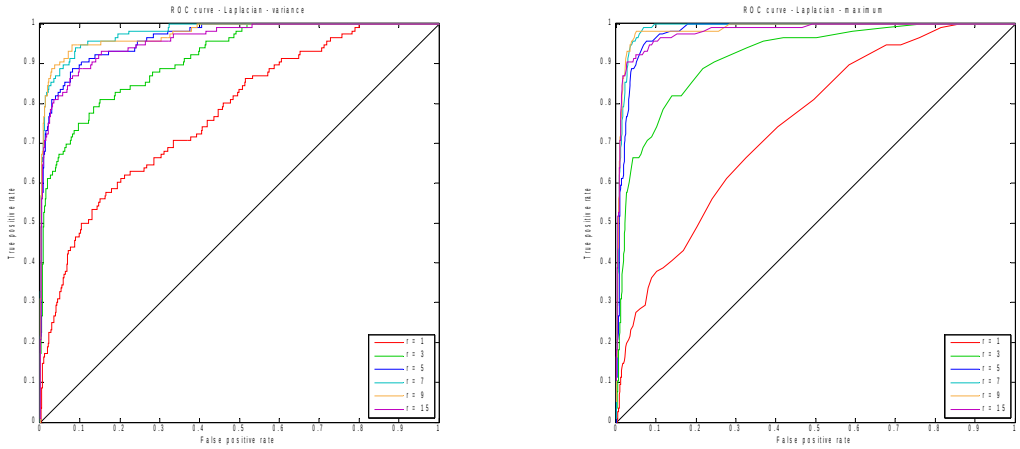


Fig. 31: ROC curves for Laplacian transformation features – measurements variance and maximum for r = {1,3,5,7,9,15}

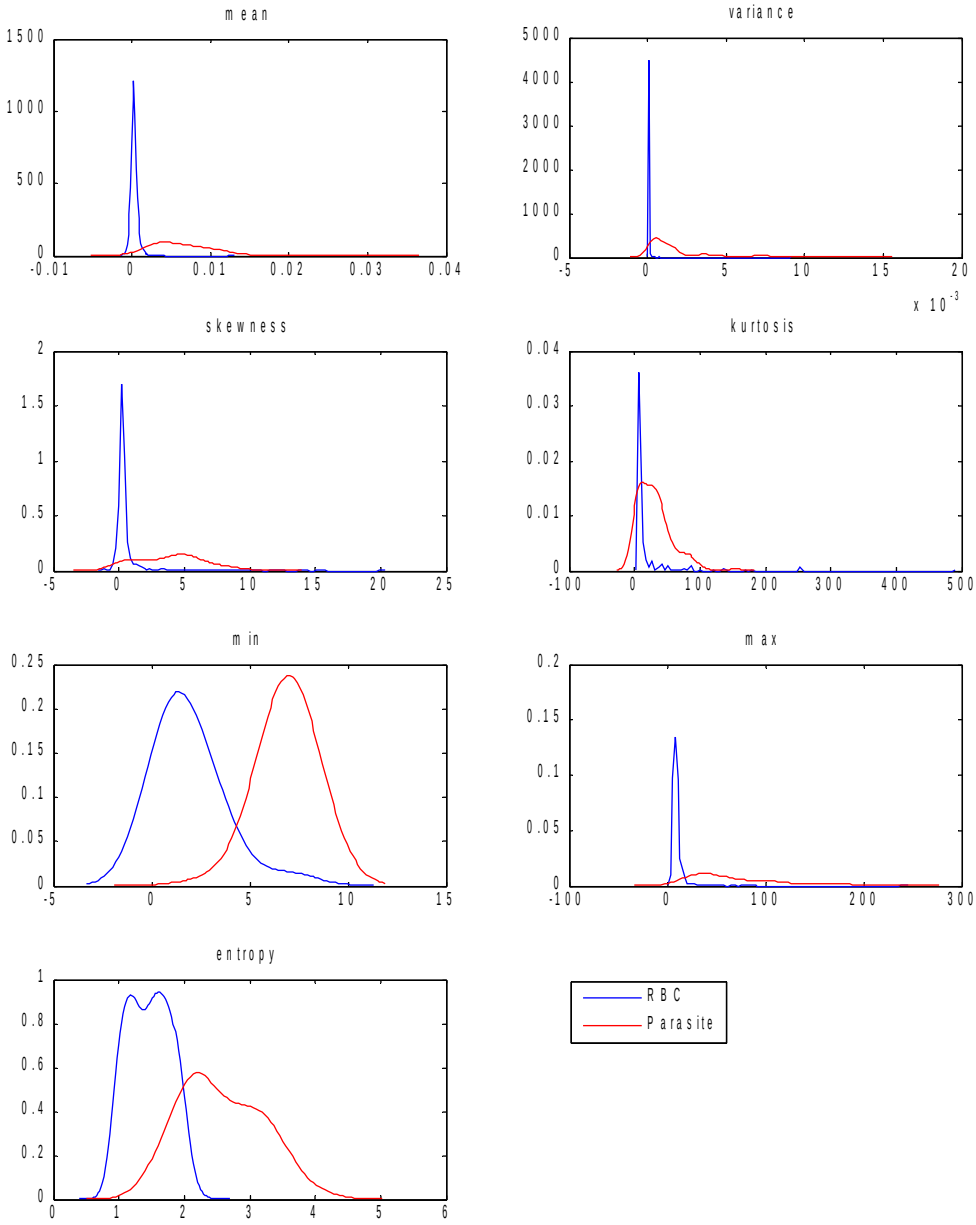


Fig. 32: Estimated probability density functions for flat texture features for the size of the median operator window r = 18

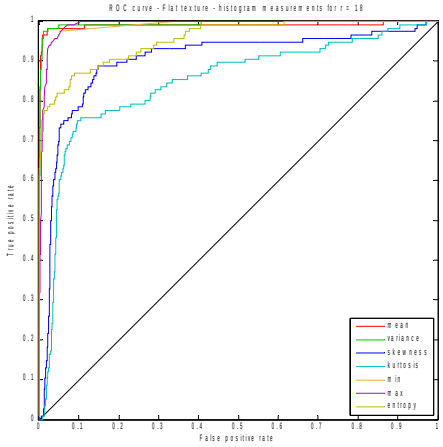


Fig. 33: ROC curves for flat texture features for the size of the median operator window r = 18

Appendix B – List of Project Files

This appendix briefly describes Matlab functions, scripts, and other files distributed as an integral part of this work. The main functions and scripts are set in bold. The rest of the functions are usually supplementary or utility functions, but some of them may be found useful in various other tasks

All Matlab functions and scripts were created solely by the author of this thesis. Blood smear images and their textual descriptions were obtained from the Public Health Image Library of the Centers for Disease Control and Prevention [35]. All functions contain Matlab style help describing the syntax and use of the function. Each function and script has been thoroughly commented to describe and clarify the implemented method and make the code easily readable and understandable.

`featuresExtract/`
Project files for the feature evaluation part of the project.

`eval/`
Feature performance evaluation scripts.

`evalCooccurrence.m`
Script for evaluation of the co-occurrence features.

`evalCooccurrence.mat`
Data file with all calculated co-occurrence features.

`evalFlatTexture.m`
Script for evaluation of the flat texture features.

`evalFlatTexture.mat`
Data file with all calculated flat texture features.

`evalGrad.m`
Script for evaluation of the gradient transformation features.

`evalGrad.mat`
Data file with all calculated gradient transformation features.

`evalHist.m`
Script for evaluation of the histogram features.

`evalHist.mat`
Data file with all calculated histogram features.

`evalHu.m`
Script for evaluation of the Hu invariant moment features.

`evalHu.mat`
Data file with all calculated Hu invariant moment features.

`evalHu-falciparum.mat`
Data file with all calculated Hu invariant moment features for a data subset including only non-infected red blood cells and cells infected by Plasmodium falciparum parasites.

`evalLaplacian.m`
Script for evaluation of the Laplacian transformation features.

`evalLaplacian.mat`
Data file with all calculated Laplacian transformation features.

`evalRunLength.m`
Script for evaluation of the run-length features.

`evalRunLength.mat`
Data file with all calculated run-length features.

`evalShape.m`
Script for evaluation of the relative shape measurements features.

`evalShape.mat`
: *Data file with all calculated relative shape measurements features.*

`plotFeature.m`
: *Function for plotting estimated probability density functions of the calculated features.*

`plotroccurve.m`
: *Function for plotting ROC curves.*

`roccurve.m`
: *Calculates ROC curve points and AUC values.*

`lib/`

`bestFitEllipse.m`
: *Best-fit ellipse.*

`bestFitEllipseOpt.m`
: *Calculates coefficients of the best-fit ellipse by solving non-linear data-fitting problem.*

`btnCancelWaitbar_callback.m`
: *Callback function for the cancel button of the waitbar figure.*

`centralMoment.m`
: *Central image moment.*

`colorNormalization2.m`
: *Color normalization.*

`contourLength.m`
: *Length of a contour.*

`correctNonuniformIllumination.m`
: *Non-uniform illumination correction for gray-scale images.*

`correctNonuniformIllumination3.m`
: *Non-uniform illumination correction for RGB images.*

`erodeMask.m`
: *Erosion of the binary mask.*

`extinctionImage.m`
: *Extinction image transformation.*

`fCooccurrence.m`
: *Co-occurrence matrix features.*

`fFlatTexture.m`
: *Flat texture transformation features.*

`fGradientTransform.m`
: *Gradient transformation features.*

`fHistogram.m`
: *Histogram features.*

`fHuMoments.m`
: *Hu invariant moment features.*

`filenames.m`
: *Retrieves all names of given file type from a give directory.*

`fLaplacian.m`
: *Laplacian transformation features.*

`flatTexture.m`
: *Flat texture features*

`fRelativeShapeMeasurements.m`
: *Relative shape measurement features.*

`fRunLength.m`
: *Run-length features*

`getMask.m`
: *Computes the mask image from given contour points.*

`getMatFilename.m`
: *Generates name of the mat-file for a given image filename.*

`invariantMoments.m`
: *Calculates intensity wighted invariant image moments.*

`orientation.m`
Orientation of an object.
`rawMoment.m`
General (raw) image moment.
`rlm.m`
Run-length matrix.
`rlm12.m`
Accumulated run-length matrices for horizontal and vertical directions.
`scaleImage.m`
Scales a gray-scale image to a specified number of gray levels.
`SICM.m`
Scale invariant central image moment.

`rbcSegm/`
Project files for the red blood cell segmentation method and segmentation GUI.

`algorithms/`
Red blood cell segmentation
`RBCsegmentation.m`
Red blood cell segmentation method and wrapper functions.
`RBCsegmentation1.m`
A wrapper for RBCsegmentation function called by segmentation GUI. Calls the segmentation method with default parameters.
`RBCsegmentation2.m`
A wrapper for RBCsegmentation function that calls the segmentation method with parameters: 'useGetCenters2',true.
`RBCsegmentation3.m`
A wrapper for RBCsegmentation function that calls the segmentation method with parameters: 'useAdapthisteq',true.

`gui/`
Red blood cell segmentation GUI.
`filenames.m`
Retrieves all names of given file type from a give directory.
`getAlgorithms.m`
List of registered segmentation algorithms.
`getContourColor.m`
List of contour line styles and color for individual classes.
`getDescriptions.m`
List of class labels.
`getMatFilename.m`
Generates name of the mat-file for a given image filename.
`loadImage.m`
Loads an image from a file and displays it within the axes object.
`moveImage.m`
Moves an image inside the axes object in the specified direction.
`RBCsegm.fig`
Red blood cell segmentation GUI (figure file)
`RBCsegm.m`
Red blood cell segmentation GUI (main function to be run)
`redraw.m`
Redraws image and red cells contours within the axes object.
`runAlgorithm.m`
Executes selected segmetation algorithm.

`images/`
Original blood smear images and created database files.
`descriptions/`
Textual descriptions of the original blood smear images.

2710.txt; 2720.txt; 3471.txt; 4110.txt; 4884.txt; 4905.txt;
4906.txt; 4918.txt; 4944.txt; 4975.txt; 5053.txt; 5057.txt;
5138.txt; 5140.txt; 5817.txt; 5820.txt; 5825.txt; 5830.txt;
5839.txt; 5842.txt; 5843.txt; 5844.txt; 5846.txt; 5848.txt;
5851.txt; 5855.txt; 5858.txt; 5860.txt; 5861.txt; 5863.txt;
5882.txt; 5883.txt; 5927.txt; 5929.txt; 5930.txt; 5931.txt;
5933.txt; 5935.txt; 5939.txt; 5941.txt; 5946.txt; 5963.txt

highres3/

Main dataset consisting of original images and corresponding mat-files with information a bout each segmented red blood cell

2710.mat; 2710.tiff; 2720.mat; 2720.tiff; 3471.mat; 3471.tiff;
4110.mat; 4110.tiff; 4884.mat; 4884.tiff; 4905.mat; 4905.tif;
4906.mat; 4906.tif; 4918.mat; 4918.tif; 4944.mat; 4944.tiff;
4975.mat; 4975.tiff; 5053.mat; 5053.tiff; 5057.mat; 5057.tiff;
5138.mat; 5138.tiff; 5140.mat; 5140.tiff; 5817.mat; 5817.tiff;
5820.mat; 5820.tiff; 5825.mat; 5825.tiff; 5830.mat; 5830.tiff;
5839.mat; 5839.tiff; 5842.mat; 5842.tiff; 5843.mat; 5843.tiff;
5844.mat; 5844.tiff; 5846.mat; 5846.tiff; 5848.mat; 5848.tiff;
5851.mat; 5851.tiff; 5855.mat; 5855.tiff; 5858.mat; 5858.tiff;
5860.mat; 5860.tiff; 5861.mat; 5861.tiff; 5863.mat; 5863.tiff;
5882.mat; 5882.tiff; 5883.mat; 5883.tiff; 5927.mat; 5927.tiff;
5929.mat; 5929.tif; 5930.mat; 5930.tif; 5931.mat; 5931.tif;
5933.mat; 5933.tif; 5935.mat; 5935.tif; 5939.mat; 5939.tif;
5941.mat; 5941.tif; 5946.mat; 5946.tif; 5963.mat; 5963.tif

highres3-falciparum/

Dataset containing only mature Plasmodium falciparum parasites in the infected cell class.

2710.mat; 2710.tiff; 2720.mat; 2720.tiff; 3471.mat; 3471.tiff;
4110.mat; 4110.tiff; 4884.mat; 4884.tiff; 4905.mat; 4905.tif;
4906.mat; 4906.tif; 4918.mat; 4918.tif; 4944.mat; 4944.tiff;
4975.mat; 4975.tiff; 5053.mat; 5053.tiff; 5057.mat; 5057.tiff;
5138.mat; 5138.tiff; 5140.mat; 5140.tiff; 5817.mat; 5817.tiff;
5820.mat; 5820.tiff; 5825.mat; 5825.tiff; 5830.mat; 5830.tiff;
5839.mat; 5839.tiff; 5842.mat; 5842.tiff; 5843.mat; 5843.tiff;
5844.mat; 5844.tiff; 5846.mat; 5846.tiff; 5848.mat; 5848.tiff;
5851.mat; 5851.tiff; 5855.mat; 5855.tiff; 5858.mat; 5858.tiff;
5860.mat; 5860.tiff; 5861.mat; 5861.tiff; 5863.mat; 5863.tiff;
5882.mat; 5882.tiff; 5883.mat; 5883.tiff; 5927.mat; 5927.tiff;
5929.mat; 5929.tif; 5930.mat; 5930.tif; 5931.mat; 5931.tif;
5933.mat; 5933.tif; 5935.mat; 5935.tif; 5939.mat; 5939.tif;
5941.mat; 5941.tif; 5946.mat; 5946.tif; 5963.mat; 5963.tif

lib/

correctNonuniformIllumination.m

Correction of non-uniform illumination in a gray-scale image.

drawcircle.m

Draws a single circle with given parameters into an image.

drawCircles.m

Draws circles with given centers and radii into an image.

elongation.m

Calculates elongation of an object.

fillHoles.m

Fill holes in the centers of red cells.

findMaxima3d.m

Finds a given number of maxima in a 3-D accumulator matrix so that there is at least minimum specified distance between each two maxima.

getCenters.m

Estimated center locations of red blood cells and average red blood cell radius.

`getCenters2.m`
Estimated center locations of red blood cells and average red blood cell radius. Modified version of the function that looks for second largest maxima if the radius found is too small.

`getSingleCells.m`
Calculates contours of single cells.

`getSingleCellsFromWatershed.m`
Calculates contours of single cells obtained by watershed segmentation.

`maxDist.m`
Calculates maximum distance

`mginput.m`
Graphical input from mouse or cursor.

`polygonFromContour.m`
Creates an image with a polygon inside a specified contour.

`separateCells.m`
Separates single cells from cell compounds.

`watershedDist.m`
Watershed segmentation technique based on distance transformation of the binary image.

`utilities/`

`countRBCsamples.m`
Counts red blood cell samples in a given database.

`displaySamples.m`
Displays individual red blood cells samples from a given database.

`saveCellRadiusToDbFile.m`
Appends information about estimated average cell radius to the database mat-file.

`diploma_thesis-Vit_Springl.pdf`
This report in pdf format.