# Supplementary Information - A Paper Microfluidic Cartridge for Automated Staining of Malaria Parasites with an Optically Transparent Microscopy Window

**Matthew P. Horning, Charles Delahunt, Ryan Singh, Spencer H. Garing, Kevin P. Nichols***

## Typical thick and thin smear, as manually prepared

The cartridge is intended to duplicate the utility provided by manually prepared thick and thin blood smears. An example image of such a manually prepared smear on a microscope slide is shown.

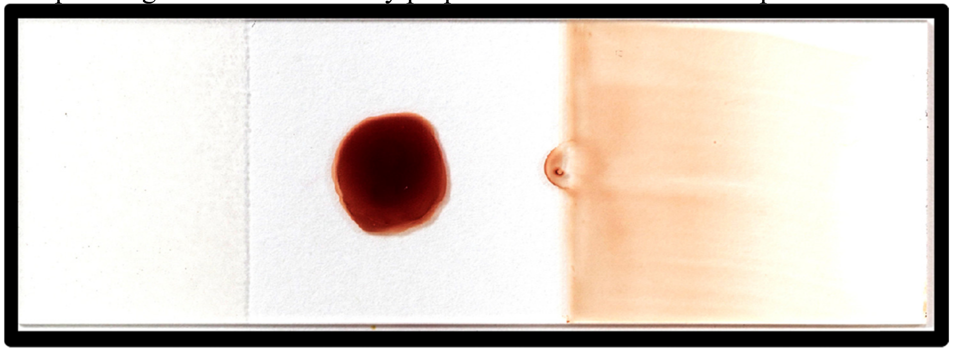

**Figure S1** – A typical thick and thin blood smear as prepared on a microscope slide by an expert microscopist. The thick smear is the circle on the left, and the thin smear is the gradient on the right.

## Optimization of stain concentration

Concentration of stain greatly affects the final image quality. Example non-optimal images (above 0.06 X) are shown.

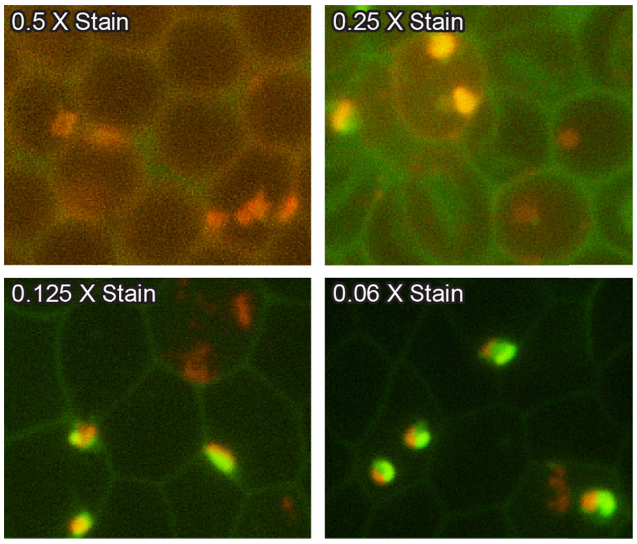

**Figure S2 –** Stain concentration is varied to reduce background signal to a level that permits clear visualization of both RNA and DNA within parasites, while still leaving the outlines of the RBC intact to permit cell counting. Stain concentration is reported as a fraction of the saturated solution described in the methods section.

## Cutting pattern for paper

The SI accessible SVG file “cutting_guide.svg,” can be used directly to duplicate the paper pattern cut using an eCraft Electronic Die Cutter (Craftwell, USA). The width of the file (the length of the long, horizontal cutting line) should be verified to be 27.94 cm. In the eCraft die cutter, paper towels were placed between two sheets of card stock, and cut with a depth of 7 (arbitrary units).

## Microscope slide adapter.

An adapter was useful to prevent the clips utilized to compress the slide from interfering with the microscope stage, and to allow the stage clamps to grip the cartridge. Figure S2 shows the adapter, as 3D printed and utilized. An STL file is included with SI to aid in duplication of this piece via external 3D printing foundries.

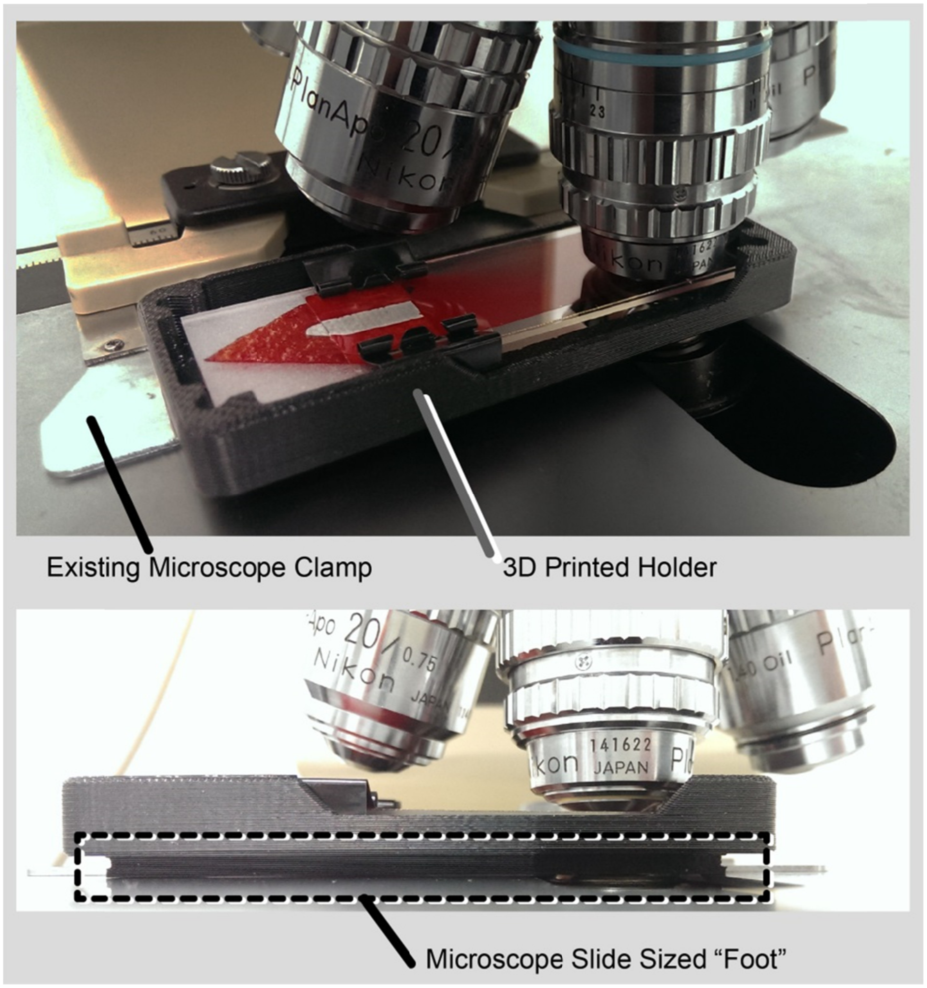

**Figure S3 –** The 3D printed slide adapter, as utilized. An STL file is available in SI to duplicate the adapter.

## MATLAB code for detection of malaria parasites in microscopy images after staining in cartridge

To duplicate the image processing algorithm utilized in this report, a new .m filed titled "countParasitesAndRBCsInAOImages.m" should be created, and the following code copied and pasted into it. There may be slight adjustments required if line breaks do not match up correctly after pasting. This script was tested on MATLAB 2013a with the image processing toolbox. Two images are also included in SI, one of which is an example image that can be directly extracted from the PDF and utilized as a test image for future code development, the other of which is the output of the script below on that image. Run the code by calling the function in MATLAB as: countParasitesAndRBCsInAOImages('filename.jpg',1)

Additionally, a more detailed technical description of the algorithm than that provided in the methods section is provided:

The image-processing algorithm has two goals: To detect parasites, and to count red blood cells (RBCs).
Parasites:

The targeted parasite signature is a bright orange blob (stained DNA) in close proximity to a bright green blob (stained RNA). Given channels R, G, and B, and pixels p, a 'bright orange' binary image is created by masking the red channel R with: {p:R(p) > Rthresh && R(p) > G(p)} = 1, where Rthresh is an adaptive threshold Rthresh = median(R) + 2*stdDev(R). Similarly, a 'bright green' binary image is created by masking the green channel G with: {p: G(p) > Gthresh && G(p) > R(p)} = 1, where Gthresh = median(G) + 2*stdDev(G). Both masked images are filtered to remove overly large blobs and single pixel noise.

To find orange and green regions that are close together, the orange masked image is dilated. Dilation is an operation on binary images in which a disc is placed on each 1-valued pixel in turn, and all pixels within the disc are converted to 1's. The effect is to increase the size of the 1-valued regions by adding a margin around them. Parasite locations are represented by the overlap of the two images: {dilated masked orange image = 1} && {masked green image = 1}.

Red blood cells:
The cell walls of RBCs stain pale green, while their interiors are typically dark. The cell walls are captured in a binary 'green' image by first histogram-equalizing the green channel G, then masking G with: {p: G(p) > localGthresh} = 1, where localGthresh = 1.1*median(G) is calculated in smaller squares that partition the image. The purpose of the partition is to minimize the effect of large green artifacts. Median filters eliminate scattered pixel noise. The image is then reversed, so that cell walls are black and cell interiors are white.

The cell interiors are separated by eroding the image. Erosion is an operation on binary images whereby a disc is placed on each {0} pixel in turn; all pixels within the disc are converted to 0's. The effect is to decrease the size of the 1-valued regions by removing their border areas; in this case cell interiors connected by noise are effectively separated. The cell interiors are then solidified (so that each cell interior is a single connected region) by dilation using a smaller disc. The number of distinct connected regions is taken to be the number of RBCs.

```
% This function counts the number of parasites and number of RBCs in an
% Acridine Orange image. It assumes that the cell walls are brighter green
% than the cell interiors. It does not distinguish between empty space and
% cell interior.
% inputs:
```

```
%      filename = image filename (including path if needed)
%      showImagesBoolean:   if True, 2 images will be shown: the original,
%                           and a greyscale with the estimated locations of
%                           parasites and RBCs. Yellow circles are parasites.
%                           A double circle implies 2 parasites close
%                           together.
%                           Red crosses are RBCs.
% outputs:
%      number of parasites, and number of RBCs.
%
% This function requires the image processing toolbox, and was tested
% with MATLAB 2013a
%
% Written by Charles Delahunt,   26 August 2013

function [numParasites, numRBCs] = countParasitesAndRBCsInAOImages(filename,
showImagesBoolean)

% load image:
[~,~,imType] = fileparts(filename);
imType = imType(2:end);
im = imread(filename,imType);

% parasites:
% method: get greenish-yellow blobs and orange blobs, then see if any are
% the right size and close to each other
% green-yellow = 150,250,0.  (image is basically 2 color)
% orange = 240,170,0

% basic processing:
imR = im(:,:,1);
imR = double(imR);
imR = imR/max(imR(:));
imG = im(:,:,2);
imG = double(imG);
imG = imG/ max(imG(:));
thrParam = 0.3; % used to set thresholds for R and G channel images

% isolate orange:
% threshold image:
thR = median (imR(:)) + thrParam*(max(imR(:)) - median(imR(:)));
imThR = imR;
imThR(imThR < thR | imR < imG) = 0;  % ie require that red > green to keep

% isolate green-yellow:
thG = median (imG(:)) + thrParam*(max(imG(:)) - median(imG(:)));
imThG = imG;
imThG(imThG < thG | imG < imR) = 0;  % ie require that green > red to keep

% We want to find green pixels (ie from imThG) that are close to orange
% pixels (ie from imThR). Do this by taking a binary image of imThR, then
% dilating it with a large disc to  give an image of pixels in the
% neighborhood of orange pixels.

temp = ones(size(imThR));
temp (imThR == 0) = 0;
temp = imerode(temp, strel('disk',2)); % kill single pixels
seC = strel('disk',25);
imDilateR = imdilate(temp, seC);

% get the overlap of the green channel and the dilated orange channel:
imGoodG = imThG;
imGoodG(imThG == 0 | imDilateR == 0) = 0;

% threshold the results in order to separate candidates:
imGoodThresh = 0.45;
imGoodG(imGoodG < imGoodThresh) = 0;

% use a binary image to count blobs. Reject small blobs
imGoodBin = zeros(size(imGoodG));
imGoodBin(imGoodG > 0) = 1;
cc = bwconncomp(imGoodBin);
paraStats = regionprops(cc,'Area', 'Centroid');

parasiteThresh = 60; % reject blobs below this size. 60 to 80 works.
paraAreas = [paraStats.Area];

numParasites = sum(paraAreas > parasiteThresh);
parasiteSizeThresh = 800;
numDoubles = sum(paraAreas > parasiteSizeThresh);
% ie this blob is too big to be just one parasite.
% detect pairs of blobs corresponding to just one parasite:
% is distance between centroids is less than 25:
paraKeep = paraStats(paraAreas > parasiteThresh);
count = 0;
for i = 1:size(paraKeep,1),
    temp = paraKeep(i).Centroid;
    for j = 1:size(paraKeep,1),
        temp2 = paraKeep(j).Centroid;
        if norm(temp - temp2) > 0 && norm(temp - temp2) < 25,
            count = count + 1;
        end
    end
end
numDuplicates = count/2;
% divide by two since each neighboring pair is counted twice.
numParasites = numParasites + numDoubles - numDuplicates; % this is output

%--------------------------------------------------------------------------

%  RBCs:

% detect cell walls:

% remove bright areas:
imG1 = imG;
thG2 = median (imG(:)) + 0.6*(max(imG(:)) - median(imG(:)));
imG1(imG > thG*0.9) = median(imG(:));

imG2 = adapthisteq(imG1);
```

```matlab
% do local median thresholding on a partition into squares of size
2*'border':
border = 20;
wallIm = zeros(size(imG2));
for i = [1+border:2*border:size(imG,1)-border,  size(wallIm,1)-border]
    for j = [1+border:2*border:size(imG,2)-border, size(wallIm,2)-border]
        temp = imG2(i-border:i+border,j-border:j+border);
        temp(temp < median(temp(:))*1.1) = 0;
        wallIm(i-border:i+border,j-border:j+border) = temp;
    end
end
 % do right hand edge:
   temp = imG2(i-border:i+border,j-border:j+border);
        temp(temp < median(temp(:))*1.1) = 0;
        wallIm(i-border:i+border,j-border:j+border) = temp;
 % do bottom edge:

% do repeated median filters to clarify image:
wallIm2 = medfilt2(wallIm);
wallIm2 = medfilt2(wallIm2);
wallIm2 = medfilt2(wallIm2);

% open the image to separate cells
wallIm3 = zeros(size(wallIm));
wallIm3(wallIm2 == 0) = 1;
seD = strel('disk',5);
wallIm3 = imopen(wallIm3, seD);
seE = strel('disk',2);
wallIm3 = imclose(wallIm3,seE);

% count the number of blobs (RBCs):
cc = bwconncomp(wallIm3);
rbcStats = regionprops(cc,'Area','FilledArea', 'FilledImage', 'Centroid');

rbcAreas = [rbcStats.Area];
rbcKeep = rbcStats(rbcAreas > 2000);  % kill small spots
numRBCs = size(rbcKeep,1); % this is an output

%--------------------------------------------------------------------

% plot results
% plot estimated cell centers:
if showImagesBoolean,
    figure, hold on,
    imshow (im), title(filename)
    figure,
    imshow(imG)
    % mark parasites:
    for i = 1:size(paraKeep),
        if paraKeep(i).Area > parasiteThresh,
            figure(2), hold on,
            xy = paraKeep(i).Centroid;
            a = paraKeep(i).Area;
            viscircles([xy(1), xy(2)], 25, 'EdgeColor', 'y');
            if a > parasiteSizeThresh,
                viscircles([xy(1), xy(2)], 35, 'EdgeColor', 'y');
            end
        end,
    end
   % mark RBCs:
    for i = 1:size(rbcKeep,1),
            xy = rbcKeep(i).Centroid;
            x(i) = xy(1);
            y(i) = xy(2);
    end
    hold on, plot(x,y,'r+')
end
    title(['Estimated numParasites = ' num2str(numParasites) '.
Estimated numRBCs = ' num2str(numRBCs)])
%----------------------------------------------------------------------
```

**Example image of parasites:**

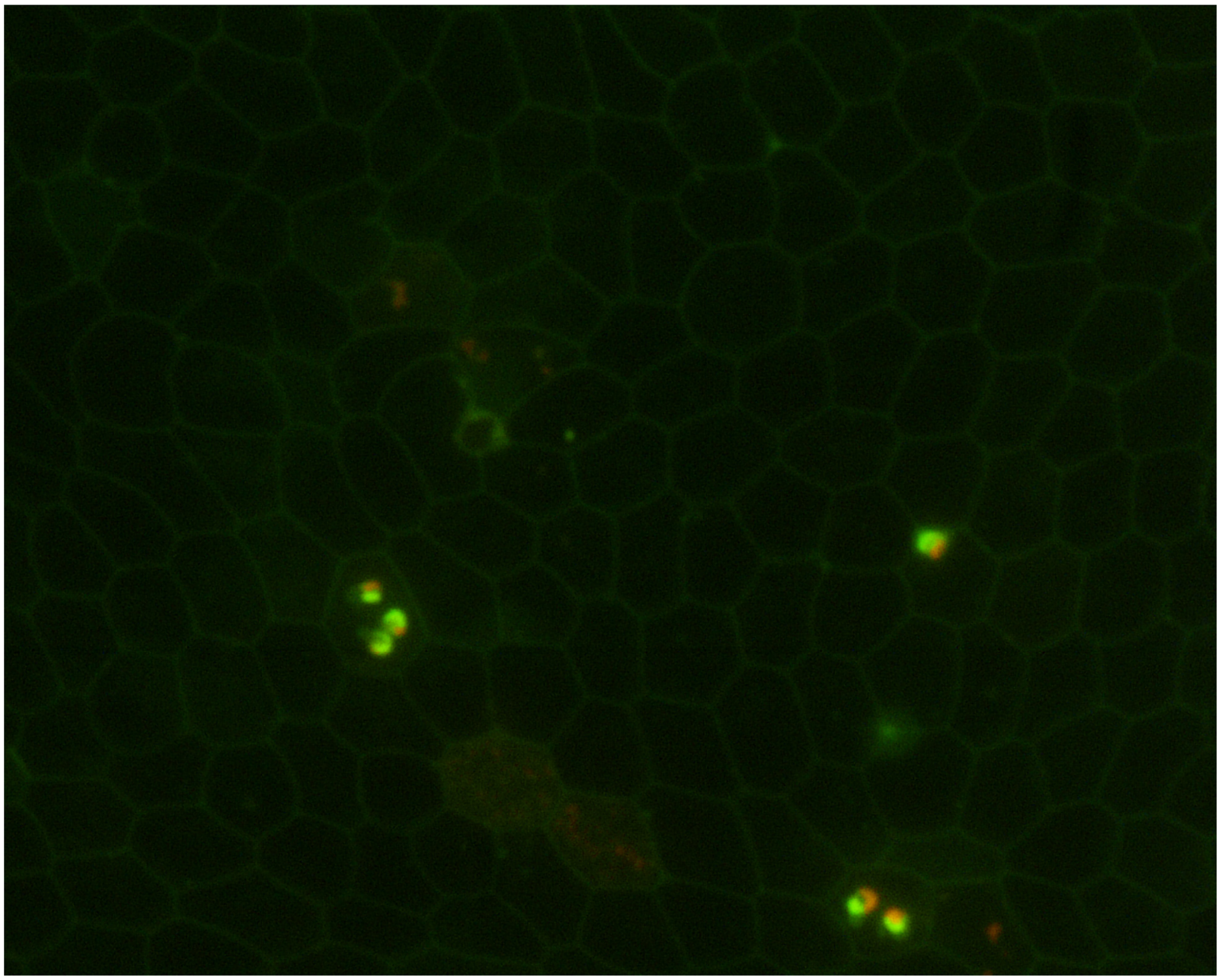

**Figure S4 -** This image can be extracted from the PDF file (either by taking a screen shot, or more directly using software such as Adobe Illustrator) and used to create a new .jpg file for testing the countParasitesAndRBCsInAOImages script with.

**Output results from countParasitesAndRBCsInAOImages script**

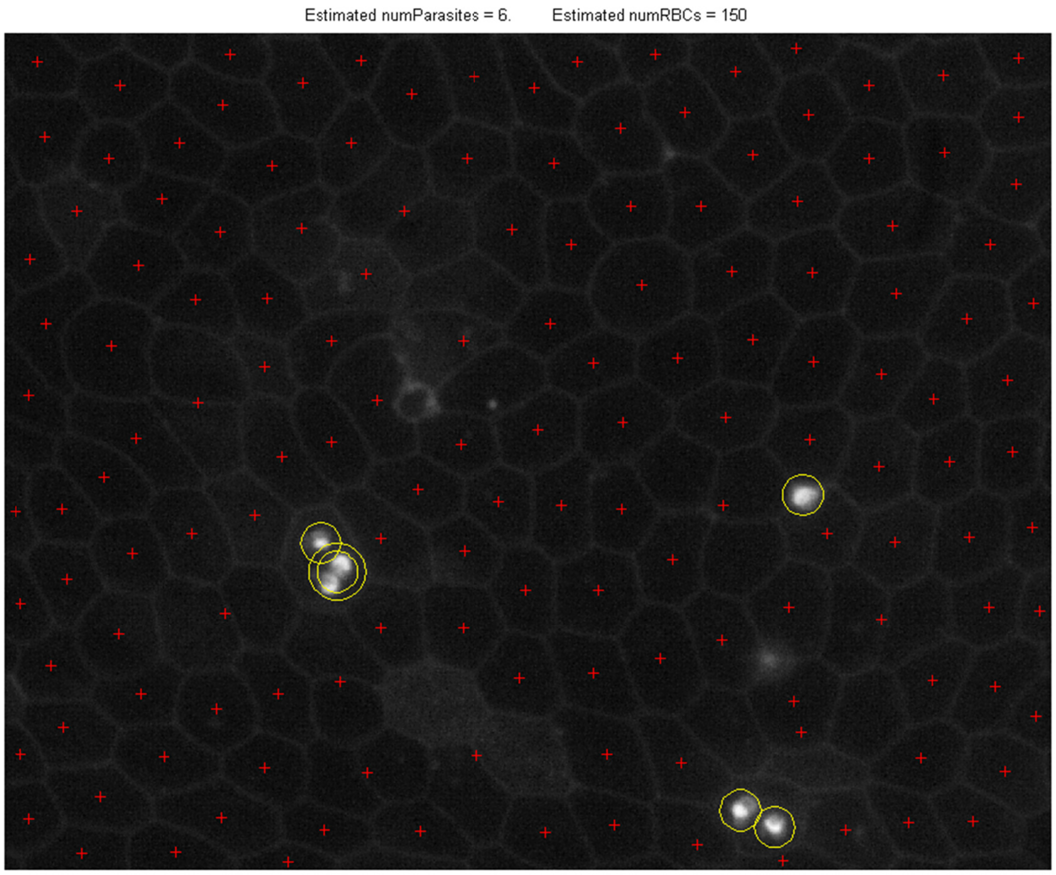

**Figure S5** - The expected output from running the countParasitesAndRBCsInAOImages script on the example image in Figure S2.