BIOM1010: Engineering in Medicine and Biology

Tutorial: Image processing for physiological measurement

Heba Khamis

Semester 2, 2018

Overview

In this tutorial, you will also apply the image processing techniques introduced in the lecture to three biomedical problems: 1) counting the number of cells in a microscope image, 2) determining the contact area made between the finger and a plate of glass, and 3) performing skin detection on an image.

Preparation

1. **Get access to MATLAB**
   - Before you attend the class it is important that you watch the YouTube videos describing the very basics of the MATLAB programming environment. There is about 1 hour 20 mins of video and some quizzes.
     - These can be found via our Moodle module (for session T2-2018) where you can also attempt simple quizzes to test your knowledge, and find other tutorial material.
     - Or you can link directly to them in this YouTube playlist
   - MATLAB is already installed on the lab computers in Room 518, but if you prefer to practice MATLAB before your tutorial, you can install a student version by following the instructions here https://www.it.unsw.edu.au/students/software/matlab.html
     OR
     - You can also access a virtual MATLAB console from your computer via UNSW MyAccess
   - Ask a tutor for help if you cannot access MATLAB.

2. **Practice the image processing techniques outlined in the lecture**
   - Before you attend the class it is important that you get some practice using the image processing techniques outlined in the lecture.
     - It is recommended that you spend some time applying the code snippets from the lecture slides to a few images to see and understand the effects that each image processing technique has.

Exercises

1. Counting the number of cells in a microscope image

The microscopy image in Figure 1 shows a field of view of fibroblast cells stained with SiR-actin (red) and Hoechst or DAPI nucleus staining (blue). One can ask: How many cells are there in this field of view? What is the average size? How much DNA is in each of the cells? How are the microtubule and actin cytoskeletons organized spatially?

Image processing and analysis provides a means to extract this information.

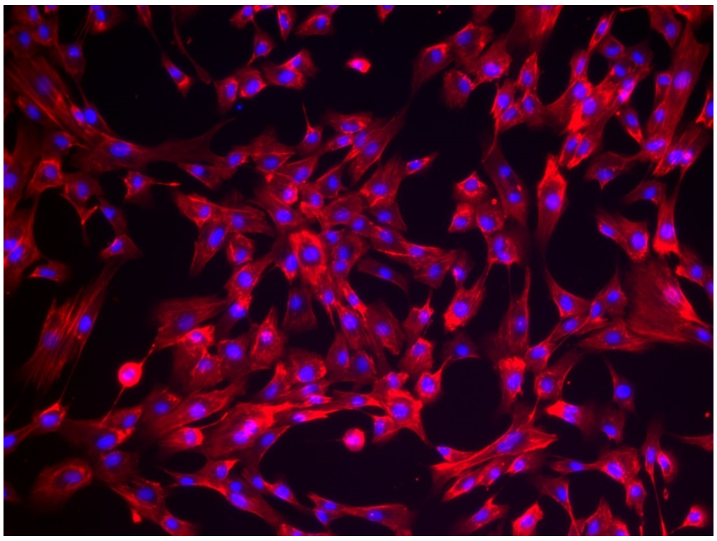

Figure 1. Microscopy field of view of fibroblast cells stained with SiR-actin (red) and Hoechst or DAPI nucleus staining (blue). [Source: http://spirochrome.com/product/sir-actin-50-nmol/]

In this exercise, we will use image processing to count the number of cells in the image in Figure 1.

- Download the file Actin_5.jpg from the course Moodle page.
- Read the image file into the MATLAB workspace.

```
rgbImage = imread('Actin_5.jpg'); % Read the image file
```

- Convert the RGB image into a grayscale image and display the grayscale image and its histogram.

```
grayImage = rgb2gray(rgbImage); % Convert RGB image to grayscale
figure; imshow(grayImage); % Display grayscale image
figure; imhist(grayImage); % Display histogram of grayscale image
```

- Display the intensities of the red component of the RGB image and its histogram.

```
rImage = rgbImage(:,:,1); % Separate red component of RGB image
figure; imshow(rImage); % Display image
figure; imhist(rImage); % Display histogram of image
```

- Also display the intensity of the green component (`gImage`) and blue components (`bImage`) and their histograms as in Figure 2.

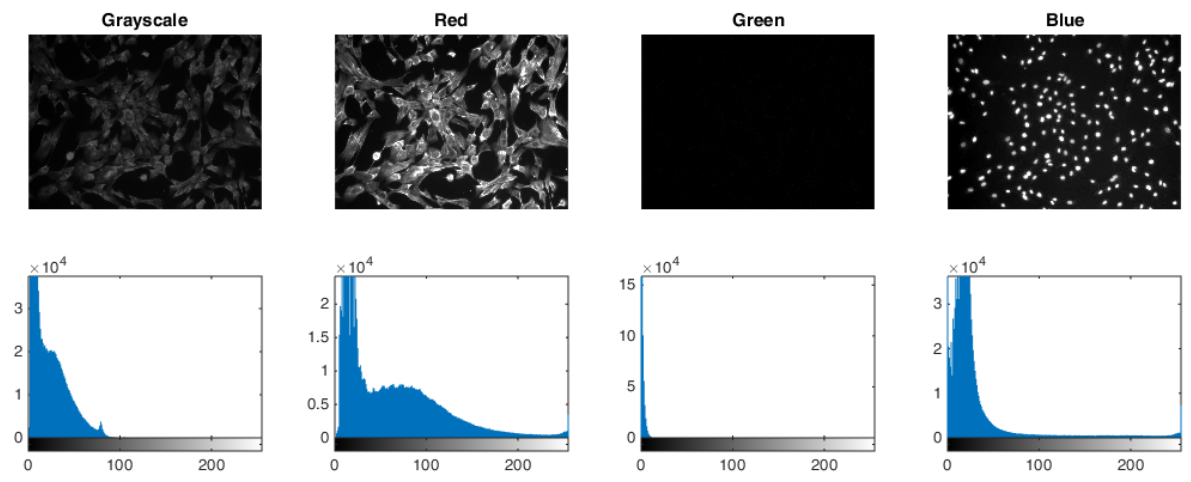

Figure 2. Grayscale, red, green and blue components of the image in Figure 1, and the respective histograms of each image.

It is clear in Figure 2 that the blue component of the RGB image can be used to identify individual nuclei and hence count the number of cells in the image.

- Determine the Otsu threshold for the image and convert the image to black and white.

```
otsuLevel = graythresh(bImage); % Get Otsu threshold
bwImage = im2bw(bImage,otsuLevel); % Convert to black & white
figure; imshow(bwImage); % Display black & white image
```

- Perform morphological closing followed by morphological opening to try to separate nuclei that are touching. The code below uses a disk-shaped structuring element with a 5 pixel radius. What is the effect of using a different shape or size for the structuring element?

```
se = strel('disk',5); % Create a disk shaped structuring element
                      % with 5 pixel radius
temp = imerode(bwImage,se); % Perform morphological erosion
nucleiImage = imclose(temp,se); % Perform morphological closing
figure; imshow(nucleiImage); % Display image
```

- Use the `regionprops` function to count the number of nuclei in the image. NB the image may contain regions that contain more than one nucleus – How can we use the shape or size properties of each region to determine how many nuclei are represented in each region?

```
% Segment the image and return the centre of mass of each region
statsCentroid = regionprops(nucleiImage,'Centroid');
% Convert result into an array
centroids = cat(1, statsCentroid.Centroid);
% Plot the nuclei image
figure; imshow(nucleiImage); hold on;
% Overlay the region centres on the image of the nuclei
plot(centroids(:,1), centroids(:,2), 'b*'); hold off;
% Display the number of regions found by region props
title(sprintf('Number of regions: %d',length(centroids)));
```

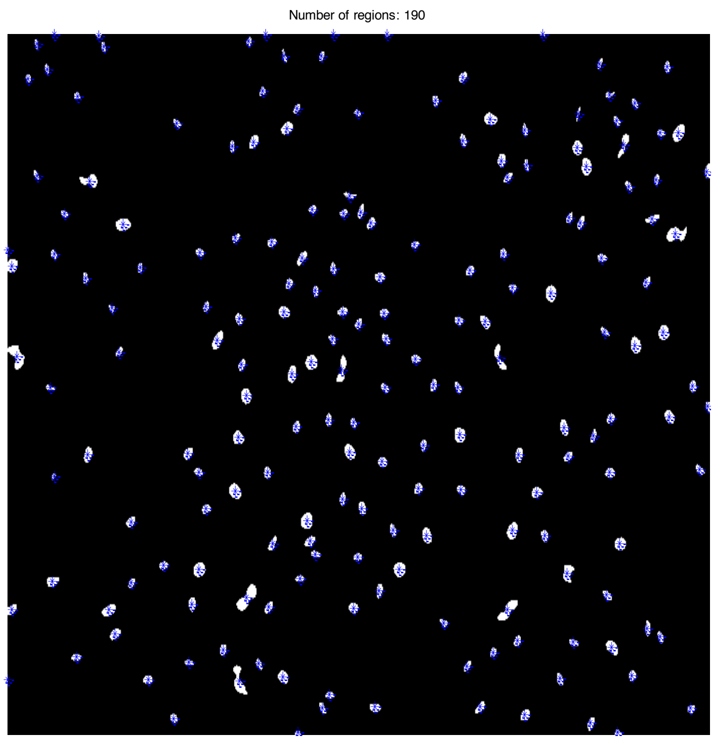

Figure 3. Final image of identified nuclei

Extension

The example microscope image above was free from artefacts and quite simple to analyse. To challenge yourself, consider the microscope field of view of red blood cells shown in Figure 4. NB: there are multiple artefacts in the image, and the colour of each red blood cell appears non-homogenous due to the characteristic biconcave shape of the cell.

Download the image file BloodCells.gif from the course Moodle page and devise an image processing protocol to count the number of red blood cells in this image.

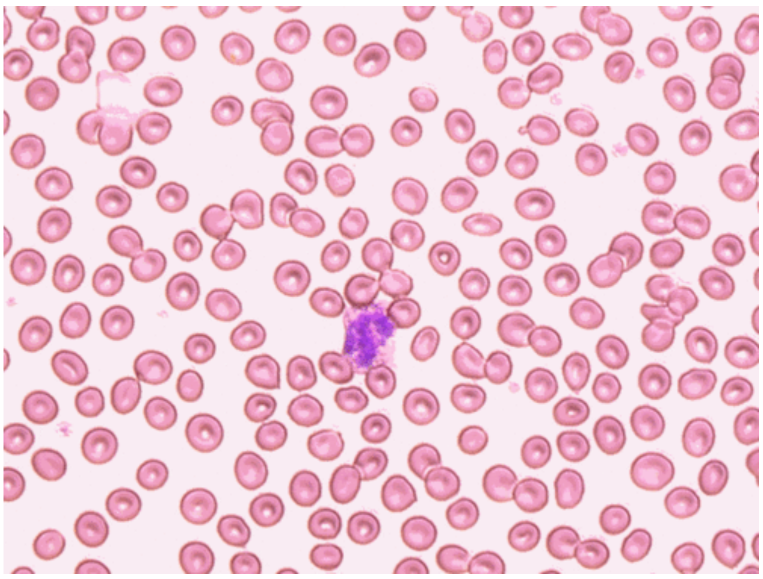

Figure 4. Microscopy field of view of red blood cells.
[Source: http://singularityhub.com/wp-content/uploads/2008/08/red-blood-cells.bmpood cells]

2. Determining the contact area of a finger on glass

A very important field in human tactile research is aimed at understanding the biomechanics of the human finger while manipulating objects. How the skin and underlying tissues deform whilst interacting with a manipulated object defines the stimuli experienced by the mechanoreceptors that send tactile information to the brain and thus the sensory feedback required for performing day-to-day manipulation tasks.

A recent approach to gaining insight into the deformation of the fingerpad involves imaging the contact area and tracking points on the fingerprint as various forces are applied. The imaging technique involves positioning a light source, camera and glass plate in such a way as to take advantage of the relative refractive indices of air, glass and skin such that light is absorbed by the skin where the fingerprint ridges are touching the glass and light is reflected by the glass where there is no skin touching. The setup is shown in Figure 5.

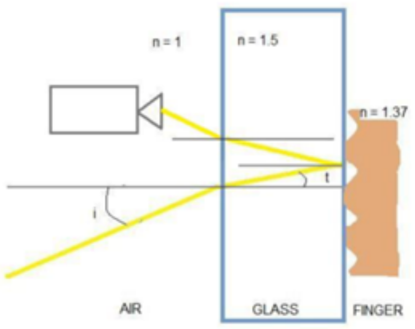

Figure 5. Setup for image capture of fingerprint in contact with glass

Before being able to track points on the fingerprint, we need to identify the region of the image that represents the contact area. In this exercise we will use image processing to determine the contact area between the finger and a glass plate.

- Using a glass microscope slide and the fluorescent ceiling lights as your light-source, position your finger on the glass as in Figure 5.
- With your smartphone, take a photo of the fingerprint that is in contact with the glass. The captured image should look similar to that in Figure 6.
- Upload the image file to your MATLAB workspace
- Read the image file into MATLAB using the `imread` function.

```
rgbImage = imread('MyFingerprint.jpg'); % Read the image file
```

- Crop the image using the interactive cropping tool so that only the fingerprint area is in the image.

```
croppedImage = imcrop(rgbImage); % Interactive cropping tool
```

- Convert the RGB image to a grayscale image and plot the image histogram.

```
grayImage = rgb2gray(rgbImage); % Convert RGB image to grayscale
figure; imshow(grayImage); % Display image
figure; imhist(grayImage); % Display histogram
```

- Perform morphological top-hat filtering, then perform contrast stretching to visualise the result. NB – you may need to try different structuring element shapes and sizes to get the desired effect

```
se1 = strel('disk',5); % Structuring element
filteredImage = imtophat(croppedImage,se1); % Top-hat filtering
figure; imshow(filteredImage); % Display figure
figure; imhist(filteredImage); % Display histogram
contrastImage = imadjust(filteredImage); % Contrast stretching
figure; imshow(contrastImage); % Display figure
figure; imhist(contrastImage); % Display histogram
```

- Perform Otsu thresholding to convert the image to black and white.

```
otsuLevel = graythresh(contrastImage); % Get Otsu threshold
bwImage = im2bw(contrastImage,otsuLevel); % Convert to black & white
figure; imshow(bwImage); % Display black & white image
```

- Perform morphological closing then opening to fill in holes. NB – you may need to try different structuring element shapes and sizes to get the desired effect.

```
se2 = strel('disk',5); % Structuring element
temp = imclose(bwImage,se2); % Morphological closing
filledImage = imopen(temp,se2); % Morphological opening
figure; imshow(filledImage); % Display image
```

- Median filter the image to soften the border. NB – you may need to try different sized median filter kernels to get the desired effect.

```
% Median filtering with 3x3 kernel
smoothedImage = medfilt2(filledImage,[3,3]);
figure; imshow(smoothedImage); % Display image
```

- Remove any residual areas. NB – you may need to try different pixel thresholds to get the desired effect.

```
% Remove residual areas having less than 50 pixels
contactImage = bwareaopen(smoothedImage,50);
figure; imshow(contactImage); % Display image
```

- The processing above should generate an image similar to that in Figure 7. Use the `regionprops` function to determine the number of pixels in the contact area.

```
statsArea = regionprops(contactImage,'Area');
area = struct2array(statsArea);
```

- NB – depending on the quality of the image some of these steps may not be needed, or alternatively, some extra processing steps may be required.

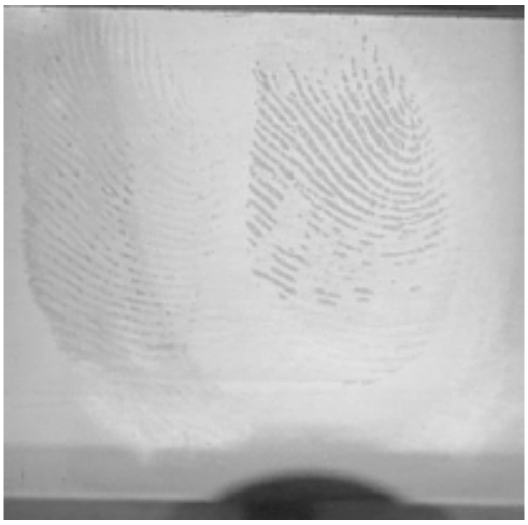

Figure 6. Image of fingerprint in contact with glass plate. Fingerprint ridges appear dark where skin absorbs the light, and light is reflected where no skin makes contact.

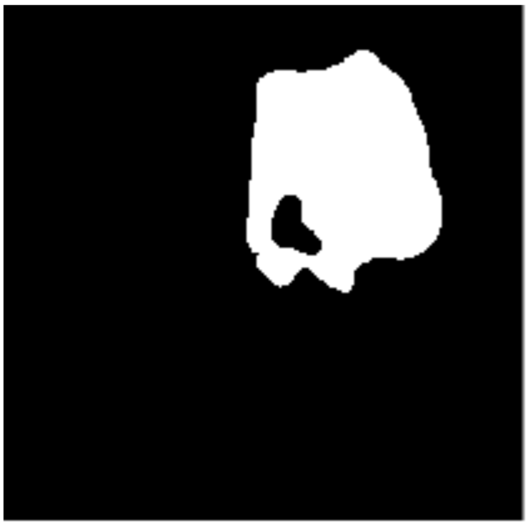

Figure 7. Black and white processed image of finger/glass plate contact area.

3. Performing skin detection

In 2006, the World Health Organization (WHO) released the Child Growth Standards for infants aged 0 to 5 years. These tables describe the population median and +/- 2 and +/- 3 standard deviations of: (1) Length/height-for-age, (2) Weight-for-age, (3) Weight-for-length, (4) Weight-for-height, (5) Body mass index (BMI)-for age, (6) Head circumference-for-age, (7) Arm circumference-for-age, (8) Triceps skinfold-for-age, and (9) Subscapular skinfold-for-age. Using these standards, a diagnosis of malnutrition can be made.

Malnutrition status in developing countries is currently assessed by arm circumference only (see Figure 8). There is however, concern about the inter- and intra-observer variability of this and other measurements. A new technique aims to use image processing techniques to measure height, head circumference and arm circumference from one or multiple images of the subject.

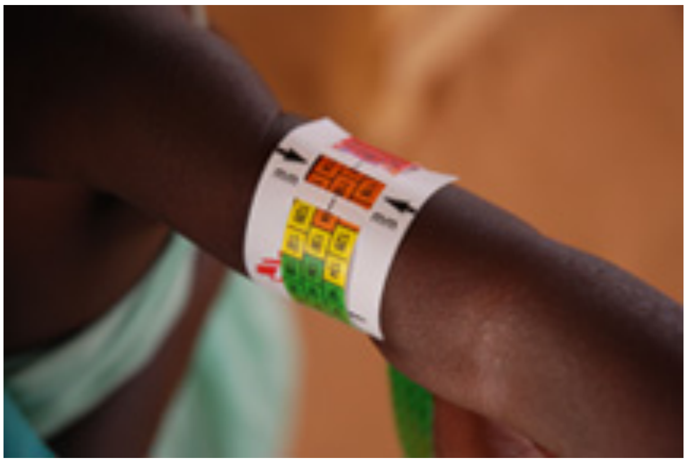

Figure 8. Current malnutrition assessment technique using a band to measure middle upper arm circumference. [Source: http://malnutrition.msf.org.au/muac.html]

One of the steps involved in determining a person's arm circumference from a photograph is identifying the arm. Skin-detection techniques can be used to localise the area of the arm. In this exercise we will perform crude skin detection based on colour matching.

- A pixel with red intensity `R`, green intensity `G` and blue intensity `B` (intensity being an 8-bit integer, i.e. ranging from 0 to 255) is identified as being skin-coloured if it satisfies the following:

R > 95 AND G > 40 AND B > 20

AND

v = [R,G,B]
(max(v) - min(v)) > 15

AND

abs(R - G) > 15 AND R > G AND R > B

- This is actually fairly easy to implement in MATLAB.

- Download/upload an image of a person from the internet into your MATLAB workspace.
- The following MATLAB code will check each pixel in an image to determine whether it is skin-coloured and generate a black and white image where white pixels represent those that were skin-coloured and black pixels represent those that were not skin-coloured in the original image.

```
image = imread('FakeTan.jpg'); % Read RGB image
R = image(:,:,1); % Red component of image
G = image(:,:,2); % Green component of image
B = image(:,:,3); % Blue component of image

condition1 = R > 95 & G > 40 & B > 20; % Condition 1
condition2 = (max(image,[],3) - min(image,[],3)) > 15; % Condition 2
condition3 = abs(R-G) > 15 & R > G & R > B; % Condition 3

% Is skin if all 3 conditions satisfied
isSkinImage = condition1 & condition2 & condition3;
% Plot original image and logical image of skin
figure;
subplot(2,1,1); imshow(image);
subplot(2,1,2); imshow(isSkinImage);
```

- Some examples of skin-detection using colour matching are shown in Figure 9.
    - It would seem that there's no such thing as too much fake tan. Can you find an example image for which the colour-matching technique fails?

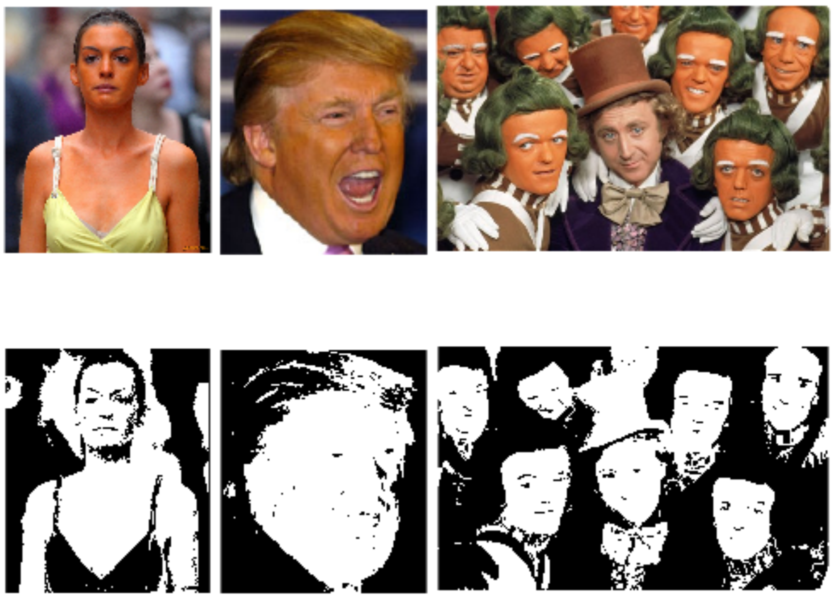
Figure 9. Examples of skin detection based on colour matching