Automated particle identification through regression analysis of size, shape and colour

J.C Rodriguez Luna[a], **J.M Cooper**[a], **S.L Neale**[a]
[a]University of Glasgow, Division of Biomedical Engineering, G12 8LT, Glasgow, Scotland.

Abstract. Rapid point of care diagnostic tests and tests to provide therapeutic information are now available for a range of specific conditions from the measurement of blood glucose levels for diabetes to card agglutination tests for parasitic infections. Due to a lack of specificity these test are often then backed up by more conventional lab based diagnostic methods for example a card agglutination test may be carried out for a suspected parasitic infection in the field and if positive a blood sample can then be sent to a lab for confirmation. The eventual diagnosis is often achieved by microscopic examination of the sample. In this paper we propose a computerized vision system for aiding in the diagnostic process; this system used a novel particle recognition algorithm to improve specificity and speed during the diagnostic process. We will show the detection and classification of different types of cells in a diluted blood sample using regression analysis of their size, shape and colour. The first step is to define the objects to be tracked by a Gaussian Mixture Model for background subtraction and binary opening and closing for noise suppression. After subtracting the objects of interest from the background the next challenge is to predict if a given object belongs to a certain category or not. This is a classification problem, and the output of the algorithm is a Boolean value (true/false). As such the computer program should be able to "predict" with reasonable level of confidence if a given particle belongs to the kind we are looking for or not. We show the use of a binary logistic regression analysis with three continuous predictors: size, shape and color histogram. The results suggest this variables could be very useful in a logistic regression equation as they proved to have a relatively high predictive value on their own.

Keywords: Cell classification, Gaussian Mixture Model, C++, Regression analysis, Diagnostics.

* J.C Rodriguez Luna, j.rodriguez-luna.1@research.gla.ac.uk

1 INTRODUCTION

Computer-aided Diagnostics(CAD) are procedures in medicine that assist in the interpretation of medical images. These techniques are commonly used in X-ray, MRI, and Ultrasound diagnostics.For example, CAD systems are used to support preventive medical check-ups in mammography(diagnostics of breast Cancer), the detection of polyps in the colon, and lung cancer. In cancer detection, for example, some computer-based techniques have been applied in the past to pigmented lesion images for investigating features to detect malignant melanoma[1]. In this paper we propose a computerized vision system for aiding in the diagnostic process of parasitic infections through logistic regression analysis.

A computer sees a image as a matrix of numbers where each element in the matrix represent one pixel. In this project we will be using extensively C++ and the OpenCV library. OpenCV(Open source Computer Vision) is a library that contains hundreds of algorithms for image and video analysis. In OpenCV the cv::Mat data structure is of special importance because it is used to manipulate images as matrices. In fact, here a image is a matrix from a computational point of view. For example, on a grey-level image the numbers in the matrix are positive 8-bit values where 0 corresponds to black and 255 to white. For a color image we have three numerical values per pixel; each one corresponding to one of the three primary colors(Red, Green, Blue). Colour images use multiple channels for each pixel [2].

For example, if we need to convert between colour representations and split the same colour image into their components channel images. Using C++ and the OpenCV library, this can be done

using the **cvtcolor** and **split** functions [2] in just a few lines of code, see Appendix A.
The goal of this project is to develop a computer vision system capable of detecting and identifying specific particles of interest as they pass through a micro fluidic channel. The algorithm must be capable of doing tracking and identification in real time. The task was divided in two main steps:

1. Real time tracking of particles.

2. Particle identification.

2 REAL TIME TRACKING OF PARTICLES

Tracking of multiple objects in video is a complex problem in computer vision. It has been approached in a variety of different ways. For example: Feature Point Tracking, Mean Shift, and Exhaustive Search. In colloidal science often a particle will have a bright centre which can be used as a recognizable feature[3, 4]. Techniques like this work well under a well controlled environment, and/or when the object we wish to track has enough stable and particular features that can be matched from frame to frame. However, in most real applications this is not the case. Take Exhaustive Search for example; In this technique a template is compared in every position using some metric. However, it is very likely the tracked object will be growing or shrinking as the object moves away or towards the camera. In other words, the appearance of the object will change just by changing the viewpoint of the object with respect to the camera. Given the huge possible changes in appearance for a single object, this technique is not the best option when dealing with bio-particle tracking. In this section I will give a very general description of the algorithm we developed to track many particles simultaneously. The C++ implementation of this algorithm allows to successfully track multiple particles at the same time even under far from ideal conditions. Object Oriented Programming (OOP) and Gaussian Mixture Model play an important role in this work.

2.1 Static Background Subtraction

The first step is to define the objects of interest to be tracked. However we will always have a background object that will be changing slightly. In an attempt to incorporate background changes Stauffer and Grimson[5] developed a algorithm that model each pixel in a frame using a mixture of Gaussian distributions(Gaussian Mixture Model).
The central idea is to fit multiple Gaussian distributions to all the previous pixel data; which includes background and foreground. This means for a given frame(m) each of the Gaussian distributions has a weight($\pi_n(i,j,m)$) depending of how frequently it has occurred in the previous frames. When considering a new frame, each point $f_n(i,j)$ is compared to the Gaussian distributions currently modeling that specific point in order to determine a single close Gaussian(in case there is any). A distribution is considered close if it is within 2.5 times the standard deviation from the mean value. In case there is no Gaussian distribution close enough, then a new Gaussian distribution is initialized to model this pixel. However, if a close Gaussian distribution is found, then that distribution is updated with this new point. Now, one of the important aspects here is that for each point $f_n(i,j)$ the largest Gaussian distributions are considered to represent the background. A pixel in a new frame is classified as foreground or background if its associated distribution is foreground or background respectively.

$$\pi_{n+1}(i,j,m) = \alpha O_n(i,j,m) + (1-\alpha) pi_n(i,j,m)$$
$$\mu_{n+1}(i,j,m) = \mu(i,j,m) + O_n(i,j,m)(\alpha/\pi_{n+1}(i,j,m)) + (f_n(i,j) - \mu_n(i,j,m))$$
$$\sigma^2_{n+1}(i,j,m) = \sigma^2_{n+1}(i,j,m) + O_n(i,j,m)(\alpha/\sigma_{n+1}(i,j,m))((f_n(i,j) - \mu_n(i,j,m))^2 - \sigma^2_n(i,j,m))$$
$$\text{where } O_n(i,j,m) = \begin{cases} 1 & \text{for the close Gaussian distribution} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where $\pi_n(i,j,m)$, $\sigma_n(i,j,m)$ and $\sigma_n(i,j,m)$ are the weighting, average and standard deviation of the m^{th} Gaussian distribution for pixel (i,j) at frame *n*.

In order to process a video sequence, we need to be able to read each one of its frames; we want to apply the same processing function to each frames of a video sequence. In our C++ implementation we do this by encapsulating our code into our own class. Then we crate a loop that will extract and process each video frame. We specify a processing function that will be called once for each frame of a video sequence. This function was defined as receiving a cv:Mat instance and outputting a processed frame of the same kind.

Fig 1: Polystyrene beads flowing in a capillary tube; this is a single frame from the input video. We will use it to illustrate background subtraction and morphological operations.

In our processing function, the part that deals with background subtraction is given in Appendix B. After applying static background subtraction to Fig.1 we get the output shown in Fig.2.

Next we apply some high-level morphological operations, see Fig.3. First we apply a closing filter to re-connect the objects that where erroneously fragmented and to fill holes. Subsequently, we apply a opening filter to remove small image noise. Like closing, opening roughly maintains the size of objects, see Appendix C.

2.2 Multiple Particle Tracking

Once the background subtraction have been successfully completed, the next step is to track individual particles. In order to do this we first find all the contours in a given frame. Next we eliminate too short or too long contours. Appendix D explain these steps. We can also compute central moments which are invariant to translation, this allows to compute the position of the shape

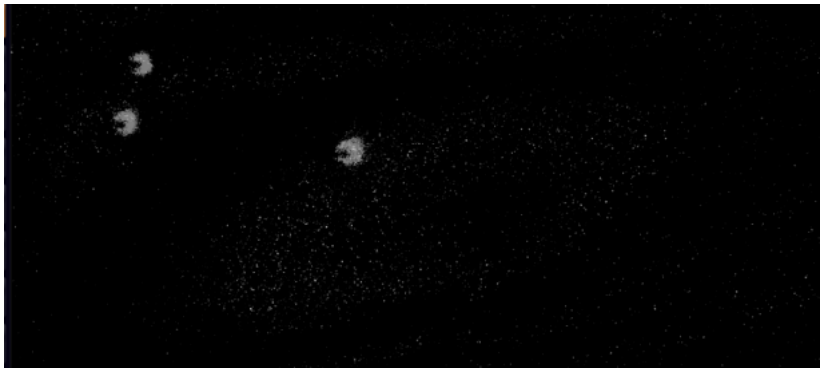


Fig 2: After applying background subtraction.

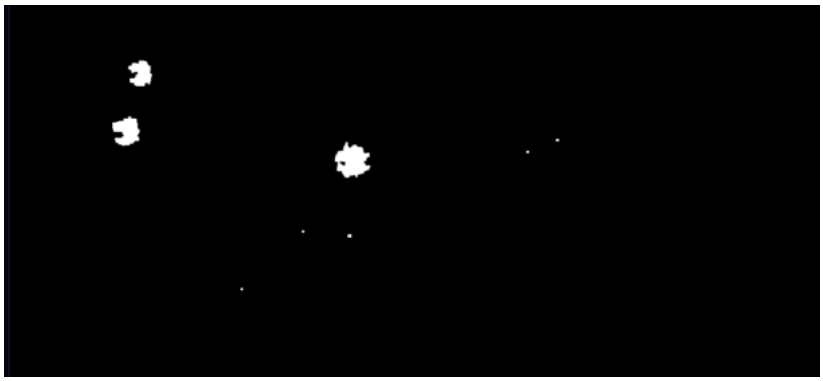


Fig 3: After applying morphological operations.

in a given frame, see Appendix E.

Now we search for the particles in the new incoming frame, see Appendix F. In order to do this, we first compute the distance from each particle in the current frame to each of the particles in the new frame. Subsequently, we update the particle's position with the position of the particle for which $dist < minDist$. Where $minDist$ is a number we pre-define taking into account how many frames per second we have in our video stream and/or how fast the particles move. The application should be considered when choosing *minDist*, for example: particle's velocity and the video camera recording frame rate. For the example shown in Fig. 7 this number was 100.0 pixel units.

A particle ID number is assigned to each **new** particle coming into the frame . We must give each particle an identity; a way to distinguish it from the others. After all these operations the particles will have a new position but the same ID number.

For the particles for which we did not find a match, **and** for those with original position inside a particular region of the frame(the far right in Fig.7) we assume they are new particles coming into the frame and we create a new particle instance for it.

Finally, we clear the vector of new incoming points before going into the next iteration. This is explained in Appendix G

Some other considerations must be taken when tracking multiple particles. For example, if two or more particles collide during the tracking process the program will eliminate those class instances and will ignore those binary shapes in the foreground image. This must be done because when

two particles collide or get too close to each other the program will not be able to make a proper differentiation.

3 PARTICLE IDENTIFICATION

The next challenge in our particle detection system is to reliably identify the particles we are looking for. We suggest to approach bio-particle detection as a classification problem, making use of one of the basic Machine Learning techniques: Binary Logistic Regression with a novel combination of particle's colour histogram, size and shape as predictors. This will be explained in more detail later on in this section.

Most machine learning problems fall in one of the two main categories:

Classification: The goal here is to predict if our data sample belongs to a certain category or not. There can be any number of predictors, and they can be discrete or continuous. However, the output of the classification algorithm is discrete (the data belongs to that category or it does not).

Regression: This technique uses a fitted multiple regression equation to predict the value of a continuous dependent variable using any number of continuous or discrete predictors.

Our problem falls in the first category. It is a classification problem, and the output of the algorithm must be a discrete number, 1 or 0 (true/false) . In other words, the algorithm must be able to *predict* with a reasonable level of confidence if a given particle belongs to the kind we are looking for or not. There are some very sophisticated algorithms that have been developed for this. For example: hidden Markov models, Naive Bayes classifiers and logistic regression. Here we will make use a Binary Logistic Regression with more than one continuous predictor.

3.1 Binary logistic regression

When making predictions in real life many different factors must be taken into account, each of which predicts with more or less reliability.

In multiple regression analysis we normally take the numerical values of two or more bits of *evidence* normally called predictors. A initial guess is made about how important each of these predictors is. Once we have our regression equation we systematically compare the prediction made by our model with the real outcome to be predicted. This systematic comparison will allow us to assess the best weight to be assigned to each one of the predictors[6]. To obtain a prediction from our model all we need to do is to take each one of the predictors and multiply it by the appropriate weight and add the resulting figures. This mathematical a analysis yields the best prediction that can be made from the evidence supplied(predictors).

Logistic regression is well suited for describing and testing a hypothesis about the relationships between a categorical dependent variable and one or more categorical or continuous predictor variables. In this short introduction to logistic regression I will consider two continuous predictors.

$$Y = \begin{cases} 1 & \text{if the particle is the one we are looking for} \\ 0 & \text{if the particle is not the one we are looking for} \end{cases} \tag{2}$$

A logistic regression model assumes the *log odds ratio* is linearly related to each one of the predictors[7]:

$$\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{3}$$

The symbol ϵ represents the error term; this symbol is used to indicate the absence of an exact relationship between Y and X_1, X_2.

It follows from Eq.3 that $p/(1-p) = exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon)$, so

$$p = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2}} \tag{4}$$

The Eq.4 gives the probability that a given particle belongs to a particular category given the values of predictors X_1, X_2. Please observe that there is no error term in Eq.4, as we do not estimate error terms in regression models. The equation would be used for values of X_1, X_2 within the range of the sample data, or perhaps only slightly outside the range. The objective is to see if past data indicates a strong enough relationship between the predictors and Y to enable future values of Y to be well predicted [7]. The rationale here is that if past values of Y can closely fit the prediction equation, then future values should similarly be closely predicted. Once $(\beta_0, \beta_1, \beta_2)$ has been obtained, these estimates can be used to describe the relationship between Y and the predictorsX_1, X_2.

The most commonly used method of estimating the parameters $(\beta_0, \beta_1, \beta_2)$ of a logistic regression model is the *method of maximum likelihood* [7]. For a sample of size n whose observations are $(y_1, y_2, \ldots, y_n)$ the corresponding random variables are $(Y_1, Y_2, \ldots, Y_n)$. Since the Y_i are assumed to be independent, the joint probability density function is

$$g(Y_1, Y_2, \ldots, Y_n) = \prod_{i=1}^{n} f_i(Y_i) \tag{5}$$

$$= \prod_{i=1}^{n} \pi_i^{Y_i}(1-\pi_i)^{1-Y_i} \tag{6}$$

since $f_i(Y_i) = \pi_i^{Y_i}(1-\pi_i)^{1-Y_i}$, where Y_i = 0,1; i = 1,2,...,n. The later is the probability that Y_i equals 0 or 1 and is thus a Bernoulli random variable relative to π_i. Eq.6 gives the probability of a particular sequence of 0's and 1's.

The *maximum likelihood estimator* (MLE) for $(\beta_0, \beta_1, \beta_2)$ is obtained by maximizing the logarithm of the likelihood function[7].

$$ln(g(Y_1, Y_2, \ldots, Y_n)) = ln[\prod_{i=1}^{n} \pi_i^{Y_i}(1-\pi_i)^{1-Y_i}] \tag{7}$$

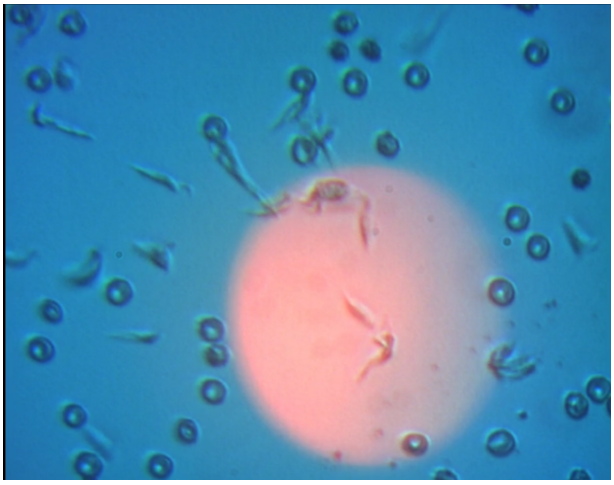
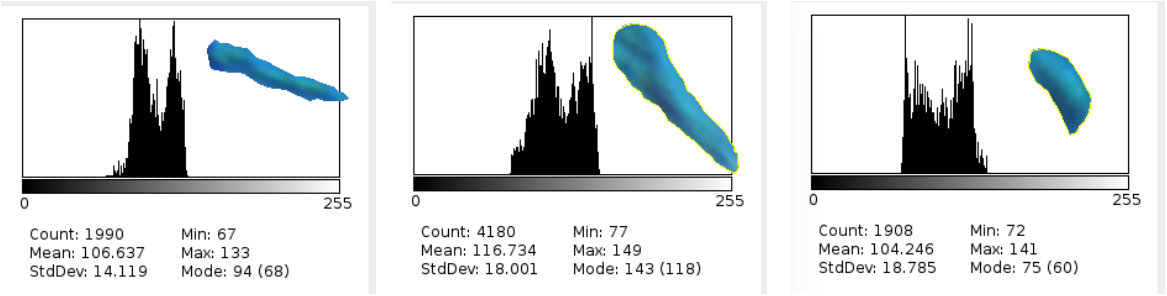


Fig 4: Image that was used to compute the histogram of red blood cells and trypanosoms[8].



(a) Trypanosoma 1 (b) Thypanosoma 2 (c) Thypanosoma 3

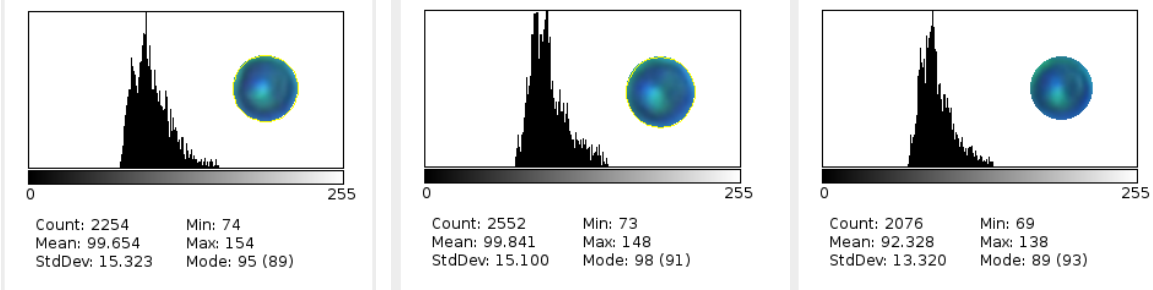
Fig 5: The histogram of three different trypanosom cells.

3.2 Colour Histogram

It is possible to extract specific features of a image by analysis of its histogram. For example, in the Fig.4 we have a sample that contains red blood cells and protozoan blood born parasites called trypanosomes[8]. Using this figure, the histogram of different cell types was obtained. The histograms for three different trypanosoms are shown in Fig.5 and the histograms for three different red blood cells are shown in Fig.6. It seems the histogram of trypanosoms is bi-modal and the histogram of red blood cells uni-modal. This represents the trypanosomes having many dark and light pixels but fewer intermediate values whereas the RBC have a preponderance of dark pixels; this kind of histogram information could be of great utility later on. Histograms constitute an effective way to characterize an image's content. This examples are not intended to prove this; their main purpose is to show the reader how a histogram looks like and their qualitative differences for different particles.

3.3 The Predictors

Area comparison To compute the area of a shape we must just count the number of pixels on that particular shape. In order to use the *area* of a particle as a predictor the algorithm will measure the area of all the particles currently being tracked in each single frame. Subsequently, the program will compute their z- score in relation with the area's normal distribution of the



(a) Red blood cell 1 (b) Red blood cell 2 (c) Red blood cell 3

Fig 6: The histogram of three different red blood cells.

targeted particle (the particle we are looking for). After that, using this z- score, the computer will estimate how rare it is to find a particle of that size(number of pixels). E.g, the metric to be used here is a probability expressed as follows:

$$P = \begin{cases} \int_z^{\infty} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz & \text{if } z > 0 \\ \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz & \text{if } z < 0 \end{cases} \tag{8}$$

Where all the observations in the *area* predictor X_1 with mean μ and variance σ have been transformed to a new set of observations of another normal random variable Z with mean 0 and variance 1 using the following transformation:

$$Z = \frac{X - \mu}{\sigma} \tag{9}$$

We must have in mind that all the values of X falling between x_1 and x_2 have corresponding Z values between z_1 and z_2, it means[9]:

$$P(x_1 < X < x_2) = P(z_1 < Z < z_2) \tag{10}$$

Histogram comparison Since we have learned that histograms constitute an effective way to characterize an image's content, it seems promising to use histogram comparison for particle identification. The idea here is to measure the similarity between two images by simply comparing their histograms. In order to use it for particle identification we will compute the histogram of all the particles currently being tracked and compare them with a histogram prototype of the particle we are looking for. A measurement function that will estimate how different, or how similar, two histograms are will be needed. There are a number of metrics that could be used to compare histograms [2]. One of these metrics is the intersection method which simply compares, for each bin, the two values in each histogram and keeps the minimum one. The similarity measure, then, is the sum of these minimum values. Another possibility is to use the Chi-Square measure which sums the normalized square difference

between the bins. Another of the metrics is the Bhattacharyya measure, which is used in statistics to estimate the similarity between to probabilistic distributions.

$$D_{\text{Correlation}}(h_1, h_2) = \frac{\sum_i (h_1(i) - \overline{h_1})(h_2(i) - \overline{h_2})}{\sqrt{\sum_i (h_1(i) - \overline{h_1})^2 \sum_i (h_2(i) - \overline{h_2})^2}} \tag{11}$$

$$D_{\text{Chi-Square}}(h_1, h_2) = \sum_i \frac{(h_1(i) - h_2(i))^2}{(h_1(i) + h_2(i))} \tag{12}$$

$$D_{\text{Bhattacharyya}}(h_1, h_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{h_1}\overline{h_2}N^2}} \sum_i \sqrt{h_1(i)h_2(i)}} \tag{13}$$

where $\overline{h_k} = \frac{sum_i(h_K(i))}{N}$

4 RESULTS AND DISCUSSION

4.1 Multiple Particle Tracking

We successfully developed a C++ program that can simultaneously track many micron-size particles using video microscopy; all is needed is a pre-recorded video. In our case these particles are 20 μm-diameter particles suspended in ID water flowing in a glass capilary tube, see Fig.7. In this figure we can see the ID number assigned to each particle as they flow from right to left. As mentioned previously, if two or more particles collide during the tracking process the program will eliminate those class instances. This algorithm works remarkably well even under tough conditions, like low contrast between foreground and background. The computation of individual trajectories, velocities and accelerations is now straightforward; this is because we have access to the ID and position of all successfully tracked particles at all times.

4.2 Particle Identification

In Section 3 we suggested to approach the challenge of particle detection as a classification problem. In this subsection we will try to assert the value of histogram and area comparison in a more quantitative way; In doing this we will also measure/study the capabilities of our tracking algorithm.

Lets start with histogram comparison. As mentioned previously, this is intended to be one of the continuous predictors, however, a metric must be defined first. In Fig. 8 we compare a single reb blood cell against ten other red blood cells and trypanosoms using three different metrics(Bhattacharyya, Chi-Square and Correlation), see Eq.11, 12,13. We can notice in Fig.8 there is a relatively high degree of differentiation(specially for the correlation metric); this suggest histogram comparison could make a reliable predictor for the logistic regression equation. Please note that we do not/should not expect a predictor to be perfect. Otherwise we would not need to make a regression analysis in the first place. In a real application we would expect to have a more noisy environment in which the differentiation will be more challenging. However, using Eq.4 will get the best prediction that can be made from the evidence supplied.

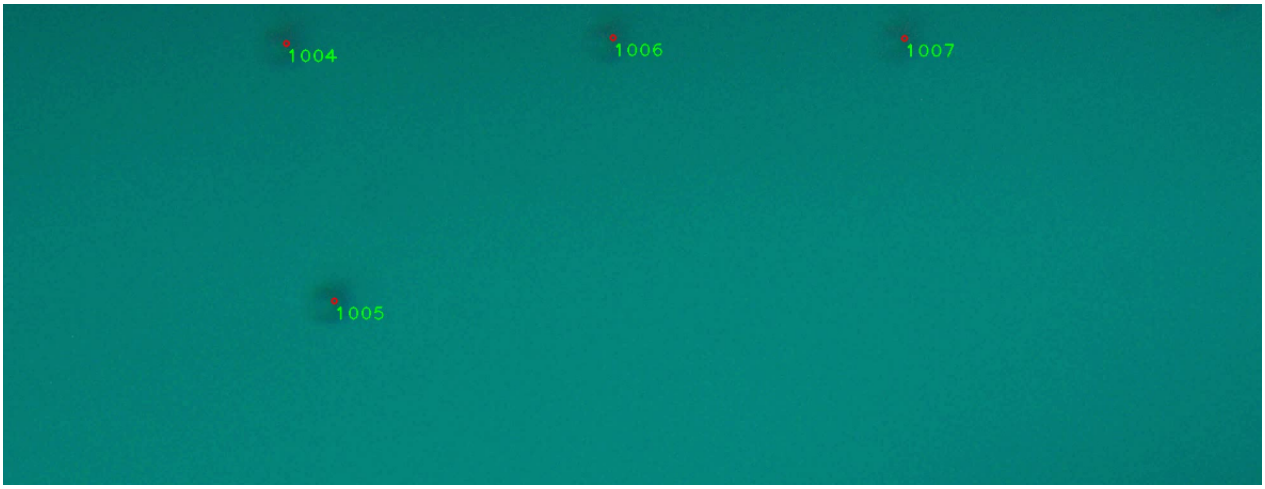
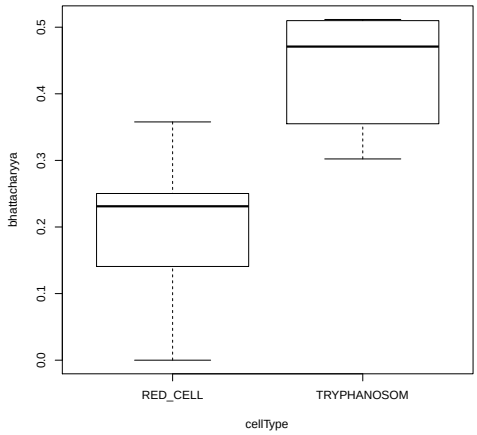
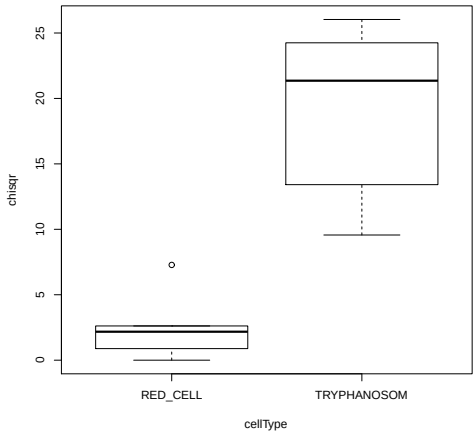


Fig 7: Polystyrene particles flowing in a capillarity tube; this is a single frame from the output video. The program successfully tracks the particles as they move through the frame. In red we can see the ID number the program assigns to particles for identification; the counting starts at 1000. http://dx.doi.org/doi.number.goes.here

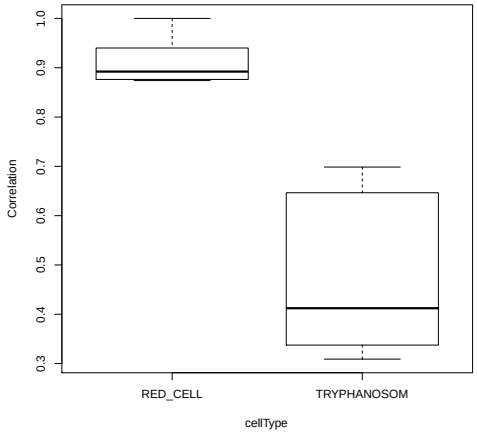
For testing the relative merit of particle size as a predictor we used our multiple-particle tracker program to compute the area of each particle in a mixture of beads; Where *area* is defined as the number of pixels in the shape. The sample is composed of beads of 5 and 8 micrometres in diameter. In Fig.9 we have the area distribution. For this particular beads sizes the two area distributions are very well defined; and we suggest the number of pixels in a shape could make a good second predictor for the logistic regression equation Eq.4.



(a) Bhattacharyya

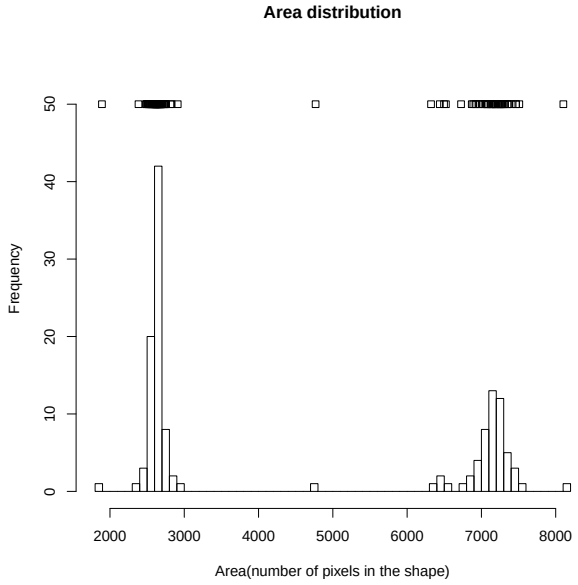


(b) Chi-Square

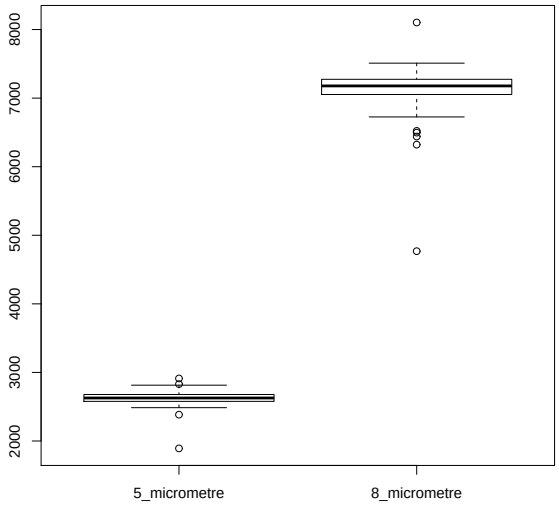


(c) Correlation

Fig 8: Three different metrics for histogram comparison. A red blood cell compared against ten other red blood cells and tryphanosoms. The relatively high degree of differentiation achieved(specially for the correlation metric) suggests histogram comparison could make a good predictor in a Logistic Regression Equation



(a) Area's histogram expressed as the number of pixels in the shape. In this mixture we have 5 and 8 micrometre diameter beads.



(b) Area distribution for the two components in the mixture.

Fig 9: To test the value of area(number of pixels in the shape) as a predictor, we also computed the area of each particle in a mixture(5 and 8 micrometre in diameter beads) flowing in a microfluidic channel. As we can see, the two area distributions are very well defined. This suggests the number of pixels in a shape could be useful as a predictor.

5 CONCLUSIONS

We successfully developed a C++ program that can simultaneously track many micron-size particles using video microscopy. The algorithm makes use of a Gaussian Mixture Model and central moments to keep track of different particles between frames. Our C++ implementation also takes into account different possible scenarios, like multiple particle collisions. For example: if two or more particles collide during the tracking process, the program will stop tracking those particles and eliminate their class instances. This program works very well even under non ideal imaging conditions, see Fig.7. The computation of individual trajectories, velocities and accelerations now becomes straightforward as we have access to the identification number (ID) and position of all successfully tracked particles at all times.

In the second part of this paper we suggests to approach bio-particle detection as a classification problem, using Binary Logistic Regression with more than one continuous predictor. To be more specific, we suggest to use colour histogram and particle size and shape to estimate the probability that a given particle belongs to a category using Eq.4. We investigated the effectiveness of particle's colour histogram and size as *predictors* using some real data, for three different metrics; the results suggest this variables could be very useful in a logistic regression equation. Future work will include corroborating this with more videos of mixtures of bio-particles and to effectively incorporate the regression equation into our multiple-tracking algorithm.

Appendix A: Colour Representations

```
cv::Mat bgr_image, grey_image; // we declare two cv::Mat data structures
cv::cvtcolor(bgr_image, grey_image, CV_BGR2GRAY); // convert bgr_image to grey-scale
    representation
std::vector<cv::Mat> bgr_images(3); // we declare a vector to store the three bgr_image
    components
cv::split(bgr_image, bgr_images); // split the bgr_image into their componet channel images
cv::Mat& blue_image = bgr_images[0]; // store the blue channel in blue_image
```

Appendix B: Extracting Foreground

```
cv::BackgroundSubstractorMOG2 mog; // The Mixture object used with all the default parameters
// extract the foreground object and convert to gray-level image
cv::mog(frame, foreground, 0.01); // where 0.01 is the learning rate
// in order to differentiate the true moving points from the background we apply band
    thresholding over the gray-level image foreground
cv::threshold(foreground, moving_points, 150, 255, cv::THRESH_BINARY);
cv::threshold(foreground, changing_points, 50, 255, cv::THRESH_BINARY);
// now we subtract the two thresholded frames to get the true foreground
cv::absdiff(moving_points, changing_points, foreground);
```

Appendix C: Morphological Operations

```
// for our particular application 6X6 structuring element works fine
cv::Mat str_el = getStructuringElement(cv::MORPH_RECT, cv::Size(6,6));
cv::morphologyEx(foreground, foreground, cv::MORPH_CLOSE, str_el); // closing filter with a 6X6
    structuring element
cv::morphologyEx(foreground, foreground, cv::MORPH_OPEN, str_el); // opening filter with a 6X6
    structuring element
```

Appendix D: Find Contours

```
// find the contour of ALL the particles
cv::findContours(foreground, // the(input) foreground image
             contours, // a vector of contours
             CV_RETR_EXTERNAL, // retrieve the external contours
             CV_CHAIN_APPROX_NONE); // all pixels of each contours

// eliminate too short or too long contours
it_c = contours.begin();
 while (it_c != contours.end()) {
      if (it_c->size() < cmin || it_c->size() > cmax)
      it_c = contours.erase(it_c);
      else
        ++it_c;
 }
```

Appendix E: Compute Central Moments

```
// get the moments
std::vector<cv::Moments> mom(contours.size());
for (int i=0; i < contours.size(); i++ )
   { mom[i] = cv::moments(contours[i], false);}

 // get the mass centers and generate the points from the new frame
 for (int j=0; j < contours.size(); j++)
   {
    commingPoints point;
    point.Xpos = (int)mom[j].m10/mom[j].m00;
    point.Ypos = (int)mom[j].m01/mom[j].m00;
```

```
    point.area = mom[j].m00;
    point.status = ``not assigned'';  // originally this point is not assigned to any previous
        point
    NewPoints.push_back(point); // generate a new instance
   }
```

Appendix F: Locate Particles

```
// search for each one of the particles new coordinates
double dist = 0.0; // auxiliary variable
itp = Particles.begin();  // iterator for the points that are actually being tracked
itpNew = NewPoints.begin(); // iterator for the points in the new frame

for(itp = Particles.begin(); itp != Particles.end(); itp++) {
  bool found = false;
  for (itpNew = NewPoints.begin(); itpNew != NewPoints.end(); itpNew++){
  dist = std::sqrt(std::pow( itp->getXpos() - itpNew->Xpos,2) + std::pow(itp->getYpos() -
      itpNew->Ypos,2));
  if (dist < minDist) {
    itp->setXpos(itpNew->Xpos); // update x position
    itp->setYpos(itpNew->Ypos); // update y position
    itp->setArea(itpNew->area); // update area
    itpNew->status = ``assigned''; // this point has been tracked(assigned to one of the
        previous ones)
    found = true;           // the tracked point has been found
  }
  }

  if (found == false) itp->notFound(); // the tracked particle has not been found
    if( itp->getNotFound() >= 2 ) {
    itp = Particles.erase(itp); // the particle is eliminated if it is not found in 2 iterations
    itp--;
    }
  }
```

Appendix G: Generate New Particles

```
// now for the particles in the new frame that were not assigned
for (itpNew = NewPoints.begin(); itpNew != NewPoints.end(); itpNew++){
// if the particle is not assigned to any point and the particle is inside a particular
// region of the frame(the beginning), then I assume is a new particle coming into the frame

 if ((itpNew->status == ``not assigned'') && (itpNew->Xpos > 2040)) {
// generate new particle instances
  Particle particle;
  particle.setXpos(itpNew->Xpos);
  particle.setYpos(itpNew->Ypos);
  particle.setArea(itpNew->area);
  particle.setID(IDnumber++);
  Particles.push_back(particle); // create the new particle instance
 }
 }
// eliminate all the detected new points in the new frame before going into the next
    iteration/frame
 NewPoints.clear();
```

References

[1] Y. Faziloglu, R. J. Stanley, R. H. Moss, W. V. Stoecker, and R. P. McLean, "Colour histogram analysis for melanoma discrimination in clinical images," *Skin Research and Technology*, 2003.

[2] K. Dawson-Howe, *A practical introduction to computer vision with opencv*. Trinity College Dublin, Ireland: Wiley, 2014.

[3] N. SL, M. M, W. JI, D. K, and K. TF, "The resolution of optical traps crated by light induced dielectrophoresis (lidep).," *Optics Express*, 2007.

[4] G. Milne, *Thesis for the degree of Doctor of Philosopy: Optical Sorting and Manipulation of Microscopic Particles*. University of St Andrews, St Andrews, Fife, Scotland.: Optical Trapping Group, 2007.

[5] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition; IEEE, Computer Society Conference*, 1999.

[6] S. Chatterjee and B. Price, *Regression Analysis by Example*. Eagle Island, Maine: Wiley, 1991.

[7] T. P. Ryan, *Modern Regression Methods*. Case Western Reserve University: Wiley, 1997.

[8] C. Kremer, C. Witte, S. L. Neale, J. Reboud, M. P. Barrett, and J. M. Cooper, "Shape-dependent optoelectronic cell lysis," *Angewandte Communications*, 2014.

[9] B. AMeery and D. ADeLancey, *Advanced Probability and Statistics, second edition*. CK-12 Fundation: Flex-Book Platform, 2012.