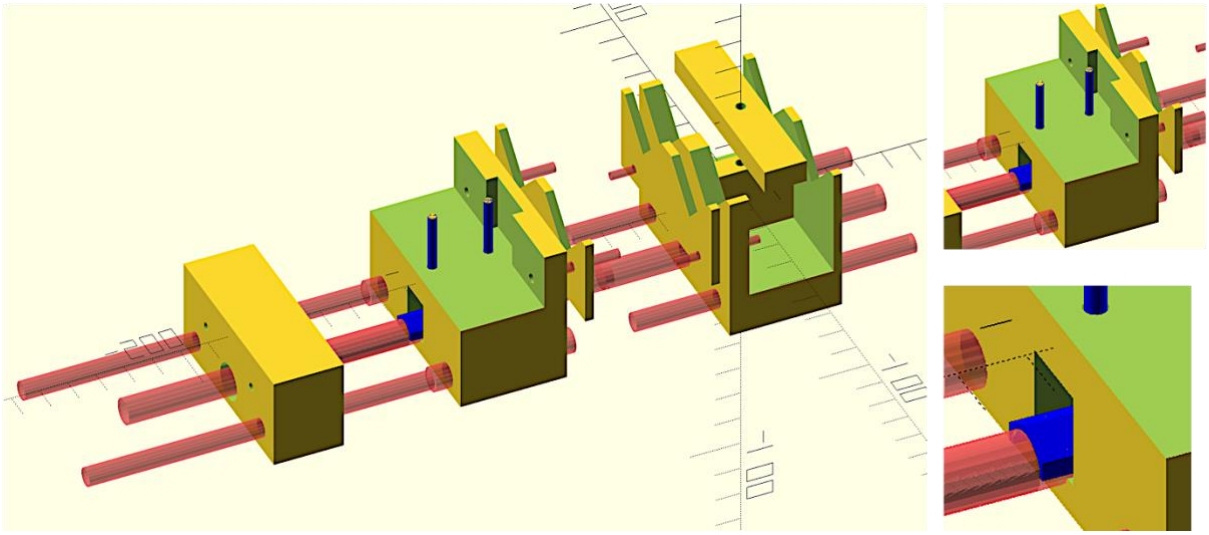
APPENDIX

1) Hardware

Syringe pump CAD model



2) Software

Below is the C ++ code of the program loaded on Arduino Nano.

```
/*
  byte vel=stateswitch[4]*4+stateswitch[5]*2+stateswitch[6]*1;
  numeri binari:
  000 = 0
  001 = 1
  010 = 2
  011 = 3
  100 = 4
  101 = 5
  110 = 6
  111 = 7
*/
#include <Stepper.h>
#include <Adafruit_SleepyDog.h>

volatile bool state = false; // when HIGH, read the states of the switches and print!
const byte pin_E = 7;
const byte pin_S0 = 3;
const byte pin_S1 = 4;
const byte pin_S2 = 5;
const byte pin_S3 = 6;
const byte pin_input = 8; // the line that reads the multiplexed state
const byte interruptPin = 2;
volatile byte stateswitch [12] ; // variabile vettore di dimensioni 12
unsigned long interrupttime = 0; // time of last interrupt press
byte S0 [ ] = {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}; // different combinations
of S0, S1, S2, S3
byte S1 [ ] = {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1};
```

```
byte S2 [ ] = {0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1};
byte S3 [ ] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1};
int dirs [] = {1, -1}; // Direzione motore
float syringetypes [] = {1, 2.5, 5, 10}; // capacità siringa utilizzata
int flowrates[] = {10, 20, 50, 100, 200, 500, 1000, 2000}; // Volume spostato
nell'intervallo, definisce Velocità
int flowrateunits [] = {1, 60}; // moltiplicatore dell'intervallo tra gli impulsi del
motore
int volumes [] = {1100, 10, 20, 50, 100, 200, 500, 1000}; // first value is practically
infinity volume
int pulses [] = { -1, 60, 120, 240}; // codificata come pausa in min. Dentro, mettere un
if che se pausa <0, esce dopo un impulso
volatile byte cont = 0; // Per loop reset
int stepsPerRevolution = 2048;  // change this to fit the number of steps per revolution
int rolePerMinute = 15;         // Adjustable range of 28BYJ-48 stepper is 0~17 rpm
int syr; // Definisce uL/passo in base a siringa utilizzata
unsigned long rip = 0; // Per ciclo mototre
float numero; // Step necessari per spostare volume desiderato
unsigned long pausa; // Pausa tra step
short dir; // Variabile utilizzata per direzione motore WD/I
unsigned long pulsetime; // Pausa tra un impulso e l'altro
unsigned long sleeptime; // Per Sleep-Mode
unsigned long breaktime; // Per pausa tra impulso e l'altro in Sleep-Mode

// initialize the stepper library on pins 9 through 12:
Stepper myStepper(stepsPerRevolution, 9, 11, 10, 12);

void setup() {
  attachInterrupt(digitalPinToInterrupt(interruptPin), sweep, CHANGE);
  pinMode(interruptPin, INPUT_PULLUP);
  // funzionamento interrupt
  pinMode(pin_input, INPUT_PULLUP);
  //pinMode(pin_input, INPUT);
  pinMode(pin_E, OUTPUT);
  pinMode(pin_S0, OUTPUT);
  pinMode(pin_S1, OUTPUT);
  pinMode(pin_S2, OUTPUT);
  pinMode(pin_S3, OUTPUT);
  // Multiplexer
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(pin_E, HIGH); //disable the MUX, just in case
  stepsPerRevolution = 16; // Dimensione step desiderato 2048 (step max giro) / 16 = 128
(360/128 = 2,8° ogni step)
  state = false;
}

void loop() {
  delay(500);
  if (state == true) {
    digitalWrite (LED_BUILTIN, LOW);
    // sweep the switches to read them
    deepswitch(); // avvio loop per lettura DIP-switch attraverso Multiplexer
    delay(10);
    // ora hai letto tutto il DIP-switch definizione variabili post lettura da var
vettoriale (stateswitch)
    int val_dir = stateswitch[0];
    dir = dirs [val_dir];
    delay(10); // debugging
    int val_flowrate = stateswitch[3] * 4 + stateswitch[4] * 2 + stateswitch[5] * 1;
//posizione nella tabella dei flow rate
    long flowrate = flowrates [val_flowrate]; // questo è il flowrate a cui fare andare la
pompa
```

```
    delay(10);
    int val_syringetype = stateswitch[1] * 2 + stateswitch[2] * 1;
    float syringetype = syringetypes  [val_syringetype]; // Tipo siringa utilizzato
    delay(10);
    int val_flowrateunit = stateswitch[6];
    int flowrateunit = flowrateunits [val_flowrateunit]; // Unità di tempo per calcolo
velocità
    delay(10);
    int val_volume = stateswitch[7] * 4 + stateswitch[8] * 2 + stateswitch[9] * 1;
    int def_volume = volumes  [val_volume]; // Volume definito della matrice
    float volume = def_volume * syringetype; // Conversione volume per tipo siringa
    delay(10);
    int val_pulse = stateswitch[10] * 2 + stateswitch[11] * 1;
    int pulse = pulses  [val_pulse]; // Tempo tra impolso e successivo
    pulsetime = pulse * 60000; // Conversione in millisecondi
    delay(10);
    // Conversione tipo siringa in uL/passo della vite
    if (syringetype == 1) syr = 16;
    if (syringetype == 2.5) syr = 78;
    if (syringetype == 5) syr = 147;
    if (syringetype == 10) syr = 208;
    delay(10);
    unsigned long intervallo = flowrateunit * 60000; // Conversione intervallo in
millisecondi
    unsigned long xen = 2048 / stepsPerRevolution * flowrate / syr; // Variabile per
definire quanti step definiti deve effettuare nell'intevallo
    pausa = intervallo / xen; // Definizione pausa tra uno step e l'altro
    float volte = volume / flowrate; // Numero di volte che viene spostato il volume
richiesto
    // all'interno della durata definita
    numero = xen * volte; // Numero di step totali
    delay(10);
    // Modalità funzionamento pompa (pulse or cost)
    if (pulse < 0) stepmotor_cost();
    if (pulse > 0) stepmotor_pulse();
    // Reset delle variabili per funzionamento cicli
    state = false;
  }
  if (state == false) { // to reset
    int sleepMS;
    digitalWrite(LED_BUILTIN, LOW);
    sleepMS = Watchdog.sleep(1000);
    digitalWrite(LED_BUILTIN, HIGH);
  }
}
void deepswitch() {
  for (int i = 0; i <= 11; i++) { // let this go to 11 if reading the entire 12-switch DIP
switch
    digitalWrite(pin_S0, S0[i]);
    digitalWrite(pin_S1, S1[i]);
    digitalWrite(pin_S2, S2[i]);
    digitalWrite(pin_S3, S3[i]);
    digitalWrite(pin_E, LOW); // enable the MUX
    delay(50);
    stateswitch [i] = digitalRead(pin_input); // records the state
    delay(50);
    stateswitch [i] = !stateswitch[i]; // inverts the output, so value=1 if switch is
selected
    digitalWrite(pin_E, HIGH); // disable the MUX
  }
}
void stepmotor_cost() { // Ciclo motore in continuo
```

```
  do {
    sei();
    myStepper.setSpeed(rolePerMinute);// Attivazione dello stepper
    // Definizione verso motore WD or I
    if (dir == -1) myStepper.step(-stepsPerRevolution);
    if (dir == 1) myStepper.step(stepsPerRevolution);
    rip++; //Contare numero step
    sleeptime = 0;
    if (pausa >= 500) { // ciclo per tenere spento il motore in caso di attesa lunga
      digitalWrite(9, LOW);
      digitalWrite(12, LOW);
      digitalWrite(10, LOW);
      digitalWrite(11, LOW);
      digitalWrite (LED_BUILTIN, LOW);
    }
    if (pausa >= 5000) { // ciclo per mandare  in sleep mode arduino se pause molto lunghe
      do { // Prima si spengono tutti i pin poi si mette in sleep
        digitalWrite(9, LOW);
        digitalWrite(12, LOW);
        digitalWrite(10, LOW);
        digitalWrite(11, LOW);
        digitalWrite (LED_BUILTIN, LOW);
        sleeptime++;
        sei();
        Watchdog.sleep(1000); // Sleep arduino
      } while (sleeptime <= pausa / 1000 && state == true);
    }
    else delay(pausa); // se pause corte resta tutto accesso compresi i pin
  } while (rip < numero && state == true); //Definizione limiti per controllare ciclo
  // Spegnimento post lavoro e reset variabili ciclo
  rip = 0;
  digitalWrite(9, LOW);
  digitalWrite(12, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite (LED_BUILTIN, LOW);
}
void stepmotor_pulse() { // Ciclo motore ad impulsi
  do {
    sei();
    myStepper.setSpeed(rolePerMinute);// Attivazione dello stepper
    // Definizione verso motore WD or I
    if (dir == -1) myStepper.step(-stepsPerRevolution);
    if (dir == 1) myStepper.step(stepsPerRevolution);
    rip++; //Contare numero step
    sleeptime = 0; // reset sleeptime
    if (pausa >= 500) { // ciclo per tenere spento il motore in caso di attesa lunga
      digitalWrite(9, LOW);
      digitalWrite(12, LOW);
      digitalWrite(10, LOW);
      digitalWrite(11, LOW);
    }
    if (pausa >= 5000) { // ciclo per mandare  in sleep mode arduino se pause molto lunghe
      do {
        digitalWrite(9, LOW);
        digitalWrite(12, LOW);
        digitalWrite(10, LOW);
        digitalWrite(11, LOW);
        digitalWrite (LED_BUILTIN, LOW);
        sleeptime++;
        sei();
        Watchdog.sleep(1000);
```

```
      } while (sleeptime <= pausa / 1000 && state == true);
    }
    else delay(pausa); // se pause corte resta tutto accesso compresi i pin
  } while (rip < numero && state == true); //Definizione limiti per controllare ciclo
  // Spegnimento post lavoro e reset variabili ciclo
  rip = 0;
  digitalWrite(9, LOW);
  digitalWrite(12, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite (LED_BUILTIN, LOW);
  for (breaktime = 0; breaktime <= pulsetime / 1000 && state == true; breaktime++) { //
Richiamo stesso void per non uscire dal loop
    sei(); // Mantenere vigile interrupt
    Watchdog.sleep(1000); // Sleep ogni 1s
  }
  if (state == true){ // Reset varaibili e richiamo loop
  breaktime = 0;
  stepmotor_pulse();
  }
}
void sweep() {
  if (millis() - interrupttime > 500) state = !state; // debouncing for interrupt
  interrupttime = millis(); // millis() relies on interrupts to count, so it will never
increment inside an ISR.
}
```

3) DOS script for plotter

Firstly, the command prompt is opened and the COM port in set in communication with the plotter in order to define the common communication parameters between the two instruments. In the case described, was set Baud rate, Parity and Data Bits of the COM2 port to which the plotter is connected through the command string, `mode com2: 9600, N, 8`.

Then, it is possible to send the file to print through a simple direct copy command to the COM2 port. However, the plotter used, due to the absence of a proper compatible program, showed to have problems with the print memory. The design sent to the plotter was too heavy causing a system error. Therefore, the file was split into two parts and the parts were sent in sequence, spaced by a short pause, in order to free the volatile memory. For this purpose, a DOS script has been developed for the automatic sending of files in sequence and repeatedly.

```
for /l %x in (1, 1, 1) do (
echo %x
copy C:\Users\Gabriele\Desktop\Disp_microfluid_1.plt COM2: /B ping 127.0.0.1 -n 5 > nul
copy C:\Users\Gabriele\Desktop\Disp_microfluid_2.plt COM2: /B ping 127.0.0.1 -n 5 > nul
)
```

To obtain more plotter passes on the platform design, was sufficient to change the third number within the brackets of the first line, indicating the desired number of the passes of the blade.

Es: `for /l %x in (1, 1, 10) do ( etc... )`

In this way, ten passages of the blade on the drawing will be obtained.

4) MATLAB script for single spheroid segmentation and fluorescence analysis

This customized script was used for analyzing images obtained from live/dead staining of HCT116 spheroids.

```
% script to measure the fluorescence of a (approx.) circular object in a
micrograph as a function of
% its distance from the center of the object.
% CHANGE in threshold search, now using only Gaussian fit of points
%
% profile and sprofile are not reset after running, so one can accumulate
the
% data and save everything together later.
%
% now it also computes the overall descriptors of the spheroid
% fluorescence, like integral, median and mean with St. Dev.
%


% now background subtraction is global. It could be made local on each ROI
% (but the image look would not be corrected)
%
% The measure function should be made to read the img from the figure data,
% not the workspace, so one can remeasure open images ...
% Children di Children di Figure, 6¬∞ elemento √® immagine e ha CData

% con shift-click che torna indietro di uno sferoide

% ora non salva su excel ma su file di testo


global img numblobs profile sdprofile absfilename f1 f  subimg
integralfluor meanfluor medianfluor objectarea thr
global statistics object hb ht
if exist('profile')
    numblobs=length(profile);
else
    profile=cell(1); %storage of fluorescence profiles
    sdprofile=cell(1);
    numblobs=0; %measured blobs
end
if exist('meanfluor')==0
    meanfluor=[];
    medianfluor=[];
    integralfluor=[];
    objectarea=[];
end


[filename,pathname]=uigetfile('*.*','Select the image
file(s)','MultiSelect','off');
% opens the dialog box,
absfilename=[pathname,filename];
img=imread(absfilename); img=double(img);
f1=figure;
set(f1,'Name',[pathname,filename])
imshow(img,[]); %displays the image
hold on
%
```

```
roi=images.roi.Rectangle;
addlistener(roi,'ROIClicked',@measure);
title('now draw a rectangular ROI around one object')
draw(roi) %interactively draws a rectangular ROI
title('in ROI:double-click to measure, Ctrl-click to save, Shift-click to
delete last')

function measure(s,evtData)
global img numblobs profile sdprofile absfilename f1 subimg integralfluor
meanfluor medianfluor objectarea thrs
global statistics object hb ht
selectionType = evtData.SelectionType;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if strcmp(selectionType,'double')
    % now read image from figure, for robustness (switching through open
    % images)
    hf=gcf;
    hc=get(hf, 'Children');
    hcc=get(hc, 'Children');
    img=hcc(end).CData; % now img should be the one that is being measured

    disp('now I measure ...')
    si=ceil(s.Position); %the location and size of the roi
    subimg=img(si(2):si(2)+si(4)-1,si(1):si(1)+si(3)-1); %-1 as I used
ceil() beforehand
    subimgM=medfilt2(subimg,[4,4]); %filtered to find threshold
    %find local threshold

    [N,X]=hist(subimgM(:),100);
    f = fit(double(X).',N.','gauss2'); %fit two gaussians, makes a cfit
object
    if f.b1>0&&f.b2>f.b1+2*f.c1 %well separated peaks
        imagetype=1;
        bkgnd=f.b1; %choice of background
        i1=min(find(X>f.b1)); i2=max(find(X<f.b2));
        im=find(N==min(N(i1:i2)));
        thr=X(im(1)); % minimum between the two distribution of points
    elseif f.b1>0&&f.b1<f.b2&&f.a1>f.a2 %first peak is the bigger
        bkgnd=f.b1; %choice of background
        thr=f.b1+2*f.c1;
        imagetype=2;
    elseif f.b2>0&&f.b1>f.b2&&f.a2>f.a1 %second peak is the bigger
        bkgnd=f.b2; %choice of background
        thr=f.b2+2*f.c2;
        imagetype=3;
    else
        imagetype=4;
        error('something unexpected')
    end

    disp(['image type = ', num2str(imagetype)])
    disp(['threshold= ',num2str(thr)])
    subimgbin=subimgM>thr;
    [L,NUM]=bwlabel(subimgbin,8);
    t=find(L==1);tf=length(t);tn=1; %start
    for j=1:NUM
        t=find(L==j);
        if length(t)>=tf, tf=length(t); tn=j; end
    end
    subimgbin=L==tn; %use only the largest domain
    subimgbin=subimgbin./max(subimgbin(:)); %perch√© se non √® il primo
dominio
    disp('segmentation done')
```

```
    % now measure the global numbers
    objectarea=[objectarea; sum(subimgbin(:))]; %objecta area in pixels
    subimg=subimg-bkgnd; subimg=(subimg>0).*subimg;% local background
correction
    object=subimgbin.*(subimg);
    integralfluor=[integralfluor; sum(object(:))];
    sdo=((sum(object(:).^2)-
(sum(object(:))^2)/sum(subimgbin(:))))/(sum(subimgbin(:))-1); %variance
    meanfluor=[meanfluor; sum(object(:))./sum(subimgbin(:)), sqrt(sdo)];
    ind=find(object~=0); % vabb√© potevo usare questo anche per quelli
soopra ...
    medianfluor=[medianfluor; median(object(ind)), iqr(object(ind))];
    stat=regionprops(subimgbin,'all');
    %now find the data for the largest object (in case more than one ...)
    m=1;
    for i=1:length(stat)
        if stat(i).Area>m; m=stat(i).Area; ind=i; end
    end
    if exist('statistics')
        statistics=[statistics, stat(ind)];
    else
        statistics=stat(ind);
    end % to store all blobs
    numblobs=numblobs+1;
    %now start erosion circle ... when to stop it?
    profile{numblobs}=[];
    sdprofile{numblobs}=[];
    flag=1;

    while 1

        newsubimgbin=imerode(subimgbin,[1 1 1;1 1 1; 1 1 1]);
        if newsubimgbin==subimgbin, break, %all object has been eroded
        else
            border=subimgbin-newsubimgbin;
            if flag==1
                [ii,jj]=find(border);
                hb(numblobs)=plot(si(1)+jj,si(2)+ii,'w.');
                flag=0;
            end
            bordervalues=border.*double(subimg);
            profile{numblobs}=[profile{numblobs};
sum(bordervalues(:))./sum(border(:))];
            sd=((sum(bordervalues(:).^2)-
(sum(bordervalues(:))^2)/sum(border(:))))/(sum(border(:))-1);
            sdprofile{numblobs}=[sdprofile{numblobs};sqrt(sd)];
            subimgbin=newsubimgbin;
        end
    end
    %disp(profile{numblobs})
    %flipping so that first point is center:
    profile{numblobs}=flip(profile{numblobs});
    sdprofile{numblobs}=flip(sdprofile{numblobs});

    disp(['measured blob #',num2str(numblobs)])
    ht(numblobs)=text(si(1),si(2),num2str(numblobs));
    set(ht(numblobs),'Color',[1 1 0])
    %
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif strcmp(selectionType,'ctrl')
    disp('now I save ...')
```

```
    %excelfilename=[absfilename,'.xlsx']; %filename to save

    maxprofilelength=0;
    for i=1:length(profile)
        tmpl=length(profile{i});
        maxprofilelength=max([maxprofilelength, tmpl]);
    end
    tmp=zeros(maxprofilelength, length(profile));
    tmps=zeros(maxprofilelength, length(profile));
    for i=1:length(profile)
        tmp(1:length(profile{i}),i)=profile{i};
        tmps(1:length(profile{i}),i)=sdprofile{i};
    end
    [filename,pathname] = uiputfile('*.xlsx', 'define the root name for
saving the results');
    if exist([pathname,filename])~=0
        disp('WARNING: older XLSX file has been deleted!')
        delete([pathname,filename])
    end
    objectnumber=(1:size(meanfluor,1))';
    meanfluor_sd=meanfluor(:,2);
    meanfluor=meanfluor(:,1);
    medianfluor_sd=medianfluor(:,2);
    medianfluor=medianfluor(:,1);
    majoraxis=[];minoraxis=[];eccentricity=[];circularity=[];perimeter=[];
    for i=1:length(statistics)
       majoraxis=[majoraxis; statistics(i).MajorAxisLength];
       minoraxis=[minoraxis; statistics(i).MinorAxisLength];
       eccentricity=[eccentricity; statistics(i).Eccentricity];
       circularity=[circularity; statistics(i).Circularity];
       perimeter=[perimeter; statistics(i).Perimeter];
    end

    spheroids=table(objectnumber, objectarea, integralfluor, meanfluor,
meanfluor_sd, medianfluor, medianfluor_sd, majoraxis, minoraxis,
eccentricity, circularity, perimeter);
    profiles=tmp;
    profiles_sd=tmps;
    spheroidsprofiles=table(profiles);
    spheroidsprofiles_sd=table(profiles_sd);

    writetable(spheroids, [pathname, filename],'Sheet','averages');
    writetable(spheroidsprofiles, [pathname, filename],'Sheet', 'fluor
profiles')
    writetable(spheroidsprofiles_sd, [pathname, filename],'Sheet', 'fluor
profiles SD')



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif strcmp(selectionType,'shift') %delete the last object
    %and check if empty, I guess
    if numblobs>=1
        delete(hb(numblobs));
        delete(ht(numblobs));
        % rollback all the results
        numblobs=numblobs-1; %back up one step and reduce the blob count
        if numblobs>0
            objectarea=objectarea(1:end-1);
            medianfluor=medianfluor(1:end-1,:);
            meanfluor=meanfluor(1:end-1,:);
            integralfluor=integralfluor(1:end-1);
            profile=profile(1:end-1);
```

```
            sdprofile=sdprofile(1:end-1);
            statistics=statistics(1:end-1);
        else
            objectarea=[];
            meanfluor=[];
            integralfluor=[];
            medianfluor=[];
            profile={};
            sdprofile={};
            statistics=statistics([]);
        end

    end

end

end
```

5) MATLAB script for single cell segmentation and fluorescence analysis

This customized script was used for analyzing images obtained from immunostaining of N9 cells.

```
%morphological characterization of N9 cell cytoskeleton micrographs
%se uno cancella delle misure, i numeri stampati sull'immagine non sono
pi√π
%corretti.

%clearvars -GLOBAL statistics*
if exist('statisticsD')==0
    statisticsD=struct;
    statisticsF=struct;
    statisticsC=struct;
    statistics2D=struct;
    statistics2F=struct;
    statistics2C=struct;
    disp('I reset the results variables')
end
global img roi statisticsD statisticsF statisticsC statistics2D
statistics2F statistics2C tableD tableF tableC ax1 ax2 ax3 ax4
global textH filenameD filenameF filenameC

% composite image already in the ws as it may come from alternative file
% collections. For now
% Image is NxNx3 or NxNx4 where 4th is brightfield
[filenameD,pathname] = uigetfile('*.tif', 'load the DAPI image');
D=imread([pathname,filenameD]);
[filenameF,pathname] = uigetfile('*.tif', 'load the FITC image');
F=imread([pathname,filenameF]);
[filenameC,pathname] = uigetfile('*.tif', 'load the Cy3 image');
C=imread([pathname,filenameC]);
img(:,:,1)=D;img(:,:,2)=F;img(:,:,3)=C;
img=double(img); %

% correzione background per piano medio
[X,Y]=meshgrid(1:size(img,2),1:size(img,1));
A=[X(:),Y(:),ones(numel(X),1)];
t=img(:,:,1);
b1 = (A'*A)\A'*t(:);
t=img(:,:,2);
```

```
b2 = (A'*A)\A'*t(:);
t=img(:,:,3);
b3 = (A'*A)\A'*t(:);
img(:,:,1)=img(:,:,1)-(b1(1).*X+b1(2).*Y+b1(3));
img(:,:,2)=img(:,:,2)-(b2(1).*X+b2(2).*Y+b2(3));
img(:,:,3)=img(:,:,3)-(b3(1).*X+b3(2).*Y+b3(3));
img=img.*(img>1);%invece che >0
% image normalization
% must normalize in the same way for all images, or it will be difficult to
% compare cells from different images (I suppose camera and lamp
intensities stay
% the same)

% first, find upper thresholds that should exclude spikes or garbage
% arbitrarily, they are defined as those which cut a fluorescence area o
% 0.2% of the global one (analogously arbitrary with a low threshold)
Is=imresize(img,[512,512]);
lowest=Is>100;
totalfluor1=sum(sum(lowest(:,:,1)));
totalfluor2=sum(sum(lowest(:,:,2)));
totalfluor3=sum(sum(lowest(:,:,3)));
t=Is(:,:,1); %tmp
i=100; %partenza
while i<max(t(:))
    tbin=t>i;
    if sum(tbin(:))<=0.002*totalfluor1, break, end
    i=i+1;
end
ut1=i;
t=Is(:,:,2); %tmp
i=100; %partenza
while i<max(t(:))
    tbin=t>i;
    if sum(tbin(:))<=0.002*totalfluor2, break, end
    i=i+1;
end
ut2=i;
t=Is(:,:,3); %tmp
i=100; %partenza
while i<max(t(:))
    tbin=t>i;
    if sum(tbin(:))<=0.002*totalfluor3, break, end
    i=i+1;
end
ut3=i;
%upper thresholds defined
%cut the spikes
t=img(:,:,1); %tmp
i=find(t>ut1); t(i)=ut1;
img(:,:,1)=t;
t=img(:,:,2); %tmp
i=find(t>ut2); t(i)=ut2;
img(:,:,2)=t;
t=img(:,:,3); %tmp
i=find(t>ut3); t(i)=ut3;
img(:,:,3)=t;

% now normalize
img(:,:,1)=img(:,:,1)./65536;
img(:,:,2)=img(:,:,2)./65536;
img(:,:,3)=img(:,:,3)./65536;

%now display ...
```

```
f1=figure;

set(f1,'Name','composite fluorescence micrograph',
'Position',[200,50,550,700])
ax1 = axes('Position',[0.02 0.02 0.96 0.7]);
scaledimg(:,:,1)=1./max(max(img(:,:,1))).*img(:,:,1);
scaledimg(:,:,2)=1./max(max(img(:,:,2))).*img(:,:,2);
scaledimg(:,:,3)=1./max(max(img(:,:,3))).*img(:,:,3);
imshow(scaledimg(:,:,[3 2 1])), yl1=ylabel('composite image');
set(yl1,'Units','normalized')
yl1.Position(1)=0;
hold on
ax2 = axes('Position',[0.05 0.72 0.25 0.25]);
imshow(img(:,:,1),[0 max(max((img(:,:,1))))])
t1=title(filenameD);
set(t1,'Interpreter','none')
set(t1,'FontSize',7)
yl2=ylabel('Ch1/DAPI');
set(yl2,'Units','normalized')
yl2.Position(1)=0;
ax3 = axes('Position',[0.35 0.72 0.25 0.25]);
imshow(img(:,:,2),[0 max(max((img(:,:,2))))])
t2=title(filenameF);
set(t2,'Interpreter','none')
set(t2,'FontSize',7)
yl3= ylabel('Ch2/FITC');
set(yl3,'Units','normalized')
yl3.Position(1)=0;
ax4 = axes('Position',[0.65 0.72 0.25 0.25]);
imshow(img(:,:,3),[0 max(max((img(:,:,3))))])
t3=title(filenameC);
set(t3,'Interpreter','none')
set(t3,'FontSize',7)
yl4=ylabel('Ch3/Cy3');
set(yl4,'Units','normalized')
yl4.Position(1)=0;

%
axes(ax1);
roi=images.roi.Polygon;
addlistener(roi,'ROIClicked',@measure);
title('now draw a polygonal ROI around one cell')
draw(roi) %interactively draws a rectangular ROI
title('in ROI:Ctrl-click to measure, Shift-click to delete last')

%
%function to evaluate on clicking in ROI
function measure(s,evtData)
title('measuring ...')
disp('entered in measure function')
global img roi statisticsD statisticsF statisticsC statistics2D
statistics2F statistics2C tableD tableF tableC ax1 ax2 ax3 ax4
global textH filenameD filenameF filenameC
selectionType = evtData.SelectionType;

if strcmp(selectionType,'ctrl')
    disp('now I measure the cell in the roi ...')
    M=roi.createMask; %the cell mask
    Ic=img.*M; %the cell's image
    %Ics=medfilt2(Ic,[4,4]); %smoothed (to find threshold)
    %now threshold the fluorescence data for each channel
    IcD=Ic(:,:,1)>graythresh(Ic(:,:,1)); IcD=bwmorph(IcD,'close');
    IcF=Ic(:,:,2)>graythresh(Ic(:,:,2)); IcF=bwmorph(IcF,'close');
```

```
    IcC=Ic(:,:,3)>graythresh(Ic(:,:,3)); IcC=bwmorph(IcC,'close'); %binary
images
    axes(ax2), imshow(IcD,[]),
    t1=title(filenameD); set(t1,'Interpreter','none'); set(t1,'FontSize',7)
    yl2=ylabel('Ch1/DAPI'); set(yl2,'Units','normalized');
yl2.Position(1)=0;
    %binary of chosen cell only
    axes(ax3), imshow(IcF,[]),
    t2=title(filenameF); set(t2,'Interpreter','none'); set(t2,'FontSize',7)
    yl3=ylabel('Ch2/FITC');set(yl3,'Units','normalized');
yl3.Position(1)=0;
    axes(ax4), imshow(IcC,[]),
    t3=title(filenameC); set(t3,'Interpreter','none'); set(t3,'FontSize',7)
    yl4=ylabel('Ch3/Cy3');set(yl4,'Units','normalized'); yl4.Position(1)=0;

    statD=regionprops(IcD,'all');
    %now find the data for the largest object (in case more than one ...)
    m=1;
    for i=1:length(statD)
        if statD(i).Area>m; m=statD(i).Area; ind=i; end
    end
    if(isempty(fieldnames(statisticsD)))
        statisticsD=statD(ind);
    else
        statisticsD=[statisticsD, statD(ind)];
    end
    %
    statF=regionprops(IcF,'all');
    %now find the data for the largest object (in case more than one ...)
    m=1;
    for i=1:length(statF)
        if statF(i).Area>m; m=statF(i).Area; ind=i; end
    end
    if(isempty(fieldnames(statisticsF)))
        statisticsF=statF(ind);
    else
         statisticsF=[statisticsF, statF(ind)];
    end
    %
    statC=regionprops(IcC,'all');
    %now find the data for the largest object (in case more than one ...)
    m=1;
    for i=1:length(statC)
        if statC(i).Area>m; m=statC(i).Area; ind=i; end
    end
    if(isempty(fieldnames(statisticsC)))
        statisticsC=statC(ind);
    else
         statisticsC=[statisticsC, statC(ind)];
    end
    % END INCLUDED MORPHOLOGICAL STATISTICS
    %
    %mark on the window to see what is done
    posx=roi.Position(1,1);
    posy=roi.Position(1,2);
    axes(ax1), hold on,
    t=text(posx,posy,num2str(length(statisticsD)),'Color','yellow');
    textH=[textH; t]; %store the text handles for the delete function
    %
    % now measure the other numbers and store in statisticsD/F/C
    %baricentro della fluorescenza di un'immagine (per il nucleo, sul DAPI)
    [X,Y]=meshgrid([1:size(img,2)],[1:size(img,1)]);
    xc=sum(sum(X.*(img(:,:,1))))./sum(sum(img(:,:,1)));
    yc=sum(sum(Y.*(img(:,:,1))))./sum(sum(img(:,:,1)));
```

```
    %matrice delle distanze di ogni punto da (xc,yc)
    X2=(X-xc).^2; Y2=(Y-yc).^2;
    R2=X2+Y2;%squared distances matrix
    R=sqrt(R2); %distances matrix (as big as image)
    % se poi N=R.*(D2>3000); % is the distance matrix for the object

    %somma delle distanze dei punti di fluorescenza
    if(isempty(fieldnames(statistics2D)))
        ind=1;
    else
        ind=length(statistics2D)+1;
    end

statistics2D(ind).meanfluorescencedistance=sum(sum(R.*Ic(:,:,1))./sum(sum(
Ic(:,:,1)));

statistics2D(ind).meanfluorescencesquareddistance=sum(sum(R2.*Ic(:,:,1)))./
sum(sum(Ic(:,:,1)));

statistics2F(ind).meanfluorescencedistance=sum(sum(R.*Ic(:,:,2)))./sum(sum(
Ic(:,:,2)));

statistics2F(ind).meanfluorescencesquareddistance=sum(sum(R2.*Ic(:,:,2)))./
sum(sum(Ic(:,:,2)));

statistics2C(ind).meanfluorescencedistance=sum(sum(R.*Ic(:,:,3)))./sum(sum(
Ic(:,:,3)));

statistics2C(ind).meanfluorescencesquareddistance=sum(sum(R2.*Ic(:,:,3)))./
sum(sum(Ic(:,:,3)));
    %integrale della fluorescenza
    statistics2D(ind).integralcellfluorescence=sum(sum(Ic(:,:,1)));
    statistics2F(ind).integralcellfluorescence=sum(sum(Ic(:,:,2)));
    statistics2C(ind).integralcellfluorescence=sum(sum(Ic(:,:,3)));
    % sovrapposizione delle fluorescenze
    statistics2F(ind).FCoverlap=sum(sum(Ic(:,:,2).*Ic(:,:,3)));
    statistics2F(ind).FCoverlapsquared=sum(sum((Ic(:,:,2).*Ic(:,:,3)).^2));
    statistics2F(ind).corrcoeff=corr2(Ic(:,:,2),Ic(:,:,3)); %correlation
coefficient
    statistics2C(ind).FCoverlap=statistics2F(end).FCoverlap;
    statistics2C(ind).FCoverlapsquared=statistics2F(end).FCoverlapsquared;
    statistics2C(ind).corrcoeff=corr2(Ic(:,:,2),Ic(:,:,3));
    % on thresholded data

statistics2D(ind).integralthresholdedcellfluorescence=sum(sum(Ic(:,:,1).*Ic
D));

statistics2D(ind).meanthresholdedcellfluorescence=sum(sum(Ic(:,:,1).*IcD)).
/sum(sum(IcD));

statistics2F(ind).integralthresholdedcellfluorescence=sum(sum(Ic(:,:,2).*Ic
F));

statistics2F(ind).meanthresholdedcellfluorescence=sum(sum(Ic(:,:,2).*IcF)).
/sum(sum(IcF));

statistics2C(ind).integralthresholdedcellfluorescence=sum(sum(Ic(:,:,3).*Ic
C));

statistics2C(ind).meanthresholdedcellfluorescence=sum(sum(Ic(:,:,3).*IcC)).
/sum(sum(IcC));
```

```
statistics2D(ind).meanthresholdedfluorescencedistance=sum(sum(R.*Ic(:,:,1).
*IcD))./sum(sum(IcD));

statistics2D(ind).meanthresholdedfluorescencesquareddistance=sum(sum(R2.*Ic
(:,:,1)))./sum(sum(Ic(:,:,1)));

statistics2F(ind).meanthresholdedfluorescencedistance=sum(sum(R.*Ic(:,:,2).
*IcF))./sum(sum(IcF));

statistics2F(ind).meanthresholdedfluorescencesquareddistance=sum(sum(R2.*Ic
(:,:,2)))./sum(sum(Ic(:,:,2)));

statistics2C(ind).meanthresholdedfluorescencedistance=sum(sum(R.*Ic(:,:,3).
*IcC))./sum(sum(IcC));

statistics2C(ind).meanthresholdedfluorescencesquareddistance=sum(sum(R2.*Ic
(:,:,3)))./sum(sum(Ic(:,:,3)));


statistics2F(ind).FCthresholdedoverlap=sum(sum((Ic(:,:,2).*IcF).*(Ic(:,:,3)
.*IcC)));

statistics2F(ind).FCthresholdedoverlapsquared=sum(sum(((Ic(:,:,2).*IcF).*(I
c(:,:,3).*IcC).^2)));

statistics2C(ind).FCthresholdedoverlap=statistics2F(end).FCthresholdedoverl
ap;

statistics2C(ind).FCthresholdedoverlapsquared=statistics2F(end).FCthreshold
edoverlapsquared;

end %end if

if strcmp(selectionType,'shift')
    statisticsC=statisticsC(1:end-1);
    statisticsD=statisticsD(1:end-1);
    statisticsF=statisticsF(1:end-1);
    statistics2C=statistics2C(1:end-1);
    statistics2D=statistics2D(1:end-1);
    statistics2F=statistics2F(1:end-1);
    disp('I deleted the last measurement')
    %axis(ax1);
    delete(textH(end)); %deletes the last text graphics
    textH(end)=[]; %removes its handle from the list


end

if strcmp(selectionType,'right')
    % saving to excel sheet
    [filename,pathname] = uiputfile('*.xlsx', 'define the file name for
saving the results');


st=rmfield(statisticsD,{'BoundingBox','SubarrayIdx','ConvexHull','ConvexIma
ge','Image','FilledImage','Extrema','PixelIdxList','PixelList','MaxFeretCoo
rdinates','MinFeretCoordinates',});
    ta=struct2table(st);
    ta2=struct2table(statistics2D);
    tableD=[ta,ta2];
    writetable(tableD, [pathname, filename],'Sheet','DAPI');
```

```
st=rmfield(statisticsF,{'BoundingBox','SubarrayIdx','ConvexHull','ConvexIma
ge','Image','FilledImage','Extrema','PixelIdxList','PixelList','MaxFeretCoo
rdinates','MinFeretCoordinates',});
    ta=struct2table(st);
    ta2=struct2table(statistics2F);
    tableF=[ta,ta2];
    writetable(tableF, [pathname, filename],'Sheet','FITC');

st=rmfield(statisticsC,{'BoundingBox','SubarrayIdx','ConvexHull','ConvexIma
ge','Image','FilledImage','Extrema','PixelIdxList','PixelList','MaxFeretCoo
rdinates','MinFeretCoordinates',});
    ta=struct2table(st);
    ta2=struct2table(statistics2F);
    tableC=[ta,ta2];
    writetable(tableC, [pathname, filename],'Sheet','Cy3');
    %also save the tables in matlab format
    save([pathname,filename(1:end-4),'mat'],'tableD','tableF','tableC')
    disp('I saved the measurements')
    % choice menu
    ch=menu('what now?','go on with this img','go on with a new img','reset
my memory and restart','quit');
    if ch==2
        MorphoCells2d
    elseif ch==3
        clearvars -GLOBAL
        clear all
        MorphoCells2d
    end


end % end save menu

 end % end function within script
```