Supporting information

A Hydrogel-Integrated Culture Device to Interrogate T Cell Activation with Physicochemical Cues

Matthew H. W. Chin[†,‡], *Michael D. A. Norman*[§], *Eileen Gentleman*[§], *Marc-Olivier Coppens*[‡,‖] *and Richard M. Day*[†,‡,*]

[†]Centre for Precision Healthcare, Division of Medicine, University College London, London, United Kingdom

[‡]Centre for Nature Inspired Engineering, University College London, London, United Kingdom

[§]Centre for Craniofacial and Regenerative Biology, King's College London, London, United Kingdom

[‖]Department of Chemical Engineering, University College London, London, United Kingdom

*Author for correspondence: r.m.day@ucl.ac.uk

Table of Contents

IMAGE PROCESSING

Images of cells and Dynabeads were taken at 48 h post-seeding using a phase contrast microscope (Zeiss Primovert) equipped with a 5-megapixel camera (Axiocam 105 color). To quantify cells and beads, images were processed using a custom algorithm written in MATLAB (version R2019a; MathWorks). The pipeline included a pre-processing stage, where non-uniform illumination correction and foreground object enhancement were performed. As cells and beads appeared as circular objects in the images, an inbuilt MATLAB circle detection function (called imfindcircles()), was utilized to identify them. The function found circles with radii within specified search ranges by applying a two-stage circular Hough transform to the image (for details, see [158]). As cells appeared bigger than Dynabeads, radii of the detected circles were used by the algorithm to distinguish between them. For two circles to touch or overlap, the distance between their centers must be less than or equal to the sum of their radii. Therefore, the algorithm was designed to loop through every detected circle and calculate their distances from the rest in order to identify cell-bead contacts. Manual counting was also performed for one of the images by placing markers on cells and beads using ImageJ.

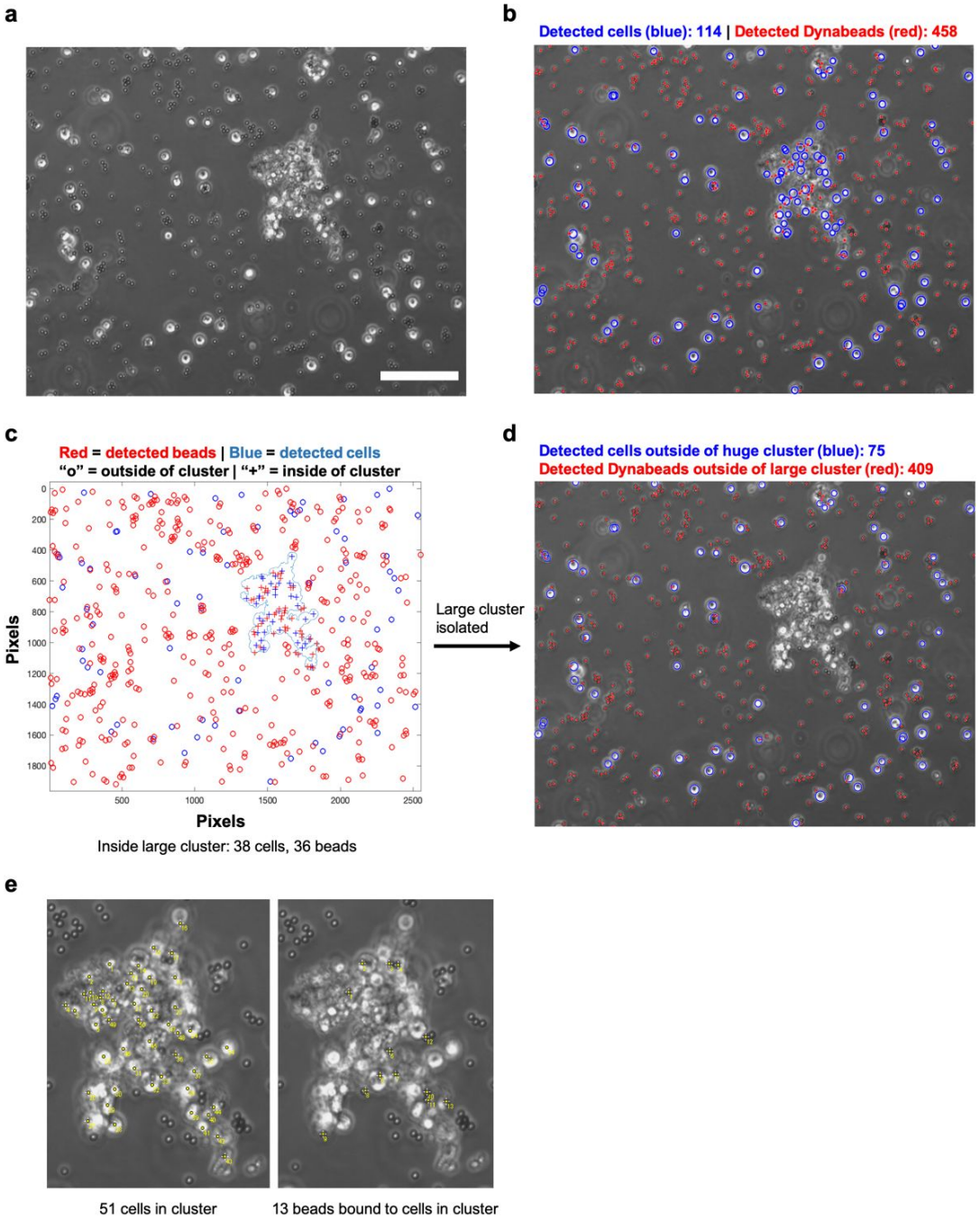

Figure S1. Image processing pipeline to analyze cell-bead interactions. Illustrated here is an example that required extra processing to quantify cells and beads, due to the presence of a large cluster. A circle detection algorithm was first applied to

the microscopy image in (**a**) to yield the results in (**b**). (**c**) The large cluster (boundary outlined in blue), however, returned some false negatives and positives (indicated by "+" markers) Cell and bead numbers detected by the algorithm are given below the plot. (**d**) The rest of the cells and beads as detected by the algorithm. (**e**) Manual counting of cells and beads inside the cluster using yellow markers.

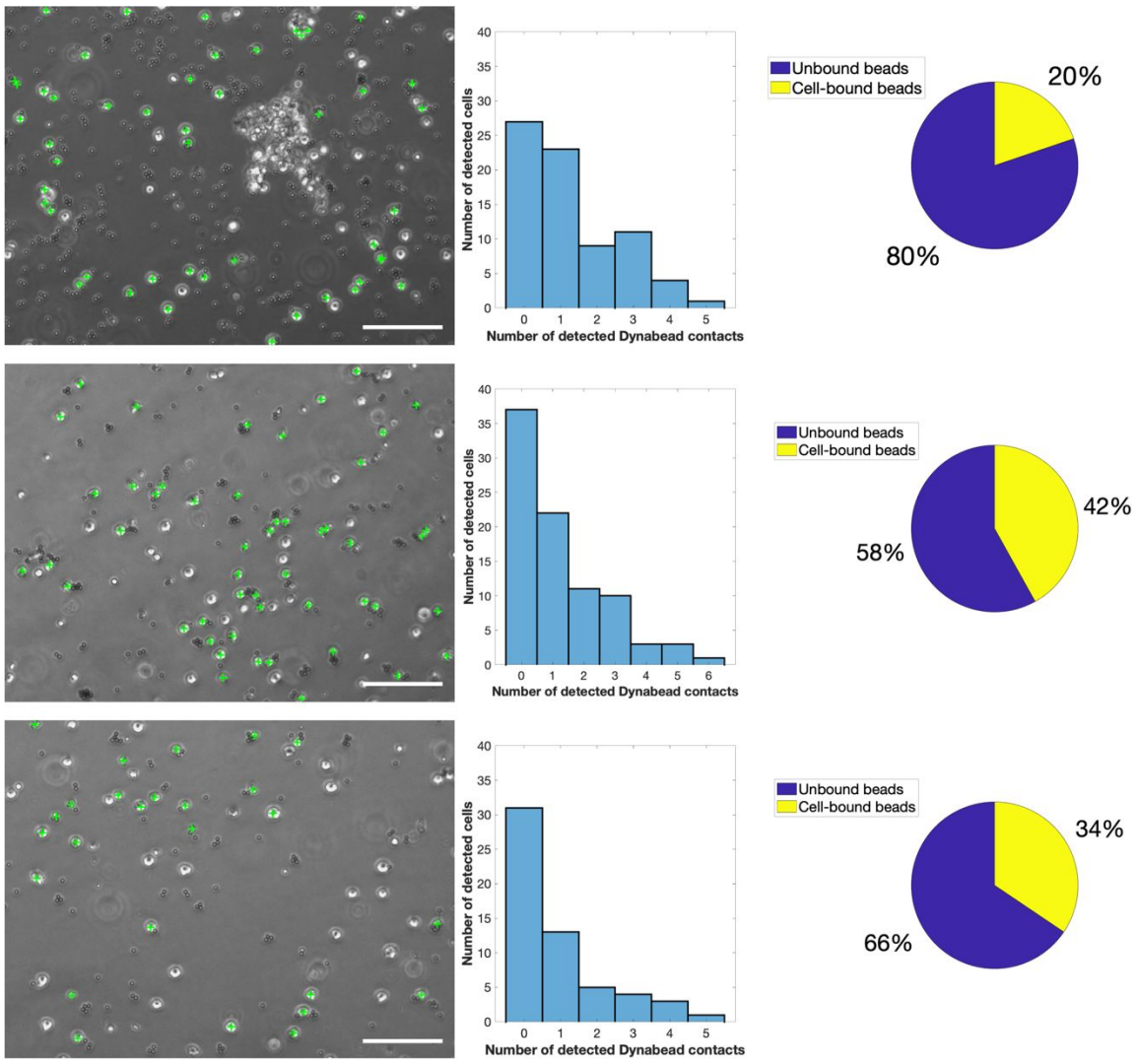

Figure S2. Detected cell-bead interactions in phase contrast microscopy images. Left column: green markers superimposed on detected bead-bound cells. Scale bar = 100 µm. Middle column: Histograms to show the number of Dynabeads bound to cells detected by the image processing pipeline. Right column: Pie charts showing the relative proportions of cell-bound beads versus unbound beads.

Quantification of cell-bead interactions. The MATLAB image processing algorithm automated and accelerated the identification of hundreds of Dynabeads and cells in microscopy images (**Figure S1** and **Figure S2**). Results returned by the algorithm showed that the majority of cells were not in contact with any beads, while some were bound to >3 beads (**Figure S2** middle column). Less than half of the detected Dynabeads were bound to at least 1 cell (**Figure S2** right column). It was noticed that one of the images contained a large cluster consisting of tightly packed cells and beads (**Figure S1a**), which led to a low accuracy of cell/bead detection in that region of the image. It was therefore decided to present a comparison of results generated from both manual counting and the algorithm (**Figure S1c, e**). Cells and beads outside of the cluster could be more accurately detected (**Figure S1d**) and the same trends were observed, where the majority of cells and beads were unbound. Nevertheless, the high proportion of unbound beads reflected suboptimal cell-bead interactions despite the fact that the beads were added at the manufacture-optimized cell-to-bead ratio of 1:1 at the start of the experiment.

```
%% Title: Script for automated particle analysis | Author: Matthew Chin

% This script is intended for the detection of cells and Dynabeads (or any
% particles that appear as circular) in phase contrast microscopy images.
% All images used were taken using a 20x objective on a Zeiss Primovert
% microscope

% Requires function RemoverOverLap.m from
% <https://uk.mathworks.com/matlabcentral/fileexchange/42370-circles-overlapremover>
% Requires function snip.m from
%   <https://uk.mathworks.com/matlabcentral/fileexchange/41941-snip-m-snipelements-out-
of-vectors-matrices>

clear all
close all
clc;

%% Load and display the image

startingFolder = pwd;

imFile = uigetfile('*.png');
rgb = imread(imFile);
whos
I = rgb2gray(rgb);
Ix_pixelLength = length(I(1,:));
xmax = Ix_pixelLength*.22;

figure('Color','w'); % Figure 1
imshow(I);
image([0 xmax],[],I)
colormap(gray(256));
title('Original Image','FontSize',18);
text(size(I,2),size(I,1)+120, ...
       'Jurkat cells w/ Dynabeads (48 hrs)', ...
       'FontSize',14,'HorizontalAlignment','right');
text(size(I,2),size(I,1)+200, ....
       'Magnificaton: 20x | 0.22 \mum per px', ...
       'FontSize',14,'HorizontalAlignment','right');

%% Correct nonuniform illumination and enhance foreground objects

% imtophat calculates the morphological opening (removes foreground) and then
% subtracts it from the original image
I2 = imtophat(I,strel('disk',20)); % Use a disk-shaped structuring element
figure('Color','w'); % Figure 2
imshow(I2);
title('Background subtracted','FontSize',18);
saveas(gcf,'Background_subtracted.png')

% Use imadjust to increase the contrast of the processed image I2
% by saturating 1% of the data at both low and high intensities and
% by stretching the intensity values to fill the uint8 dynamic range
I3 = imadjust(I2);
figure('Color','w'); % Figure 3



imshow(I3);
title('Contrast enhanced','FontSize',18);
saveas(gcf,'Contrast_enhanced.png')

%% Detect and measure Dynabeads
```

```
% By default, imfindcircles finds circular objects that are
% brighter than the background. The cells and beads already appear as
% bright circles so we don't need to do anything extra
%============================== OPTIONAL ================================%
% Gain a rough idea of the radii you are dealing with.
% imdistline creates a draggable tool that can be moved to fit across
% a cell or bead and the numbers can be read to get an approximate estimate
% of its radius

% d = imdistline;
% delete(d)
%========================================================================%

[centers1,radii1] = imfindcircles(I3,[4 14],...
       'Sensitivity',0.75,...
       'EdgeThreshold',0.15,...
       'Method','twostage');
% imfindcircles detects more circular objects
% (with both weak and strong edges) when you set the threshold
% to a lower value
% sprintf('Number of beads found: %f', length(centers1))

figure('Color','w'); % Figure 4
imshow(I3);
c1 = viscircles(centers1,radii1,'Color','g');
title("Detected Dynabeads: " + length(radii1),...
       'FontSize',18);
saveas(gcf,'Detected_beads1.png')

% Remove overlapping circles
[centers_beads,radii_beads]=RemoveOverLap(centers1,radii1,5,2);
figure('Color','w'); % Figure 5
imshow(I3);
c2 = viscircles(centers_beads,radii_beads,'Color','b');
removed = length(radii1)-length(radii_beads);
old = length(radii_beads);
title("Overlapping circles removed: "+removed,...
       'FontSize',18);
saveas(gcf,'Overlapping_circles_removed.png')

% Remove big circles that are more likely to be
% cells or false positives than beads
indices = find(abs(radii_beads)>5);
radii_beads(indices) = [];
centers_beads(indices,:) = [];
removed = old - length(radii_beads);
figure('Color','w'); % Figure 6
imshow(I3);
c3 = viscircles(centers_beads,radii_beads,'Color','r');
title("Large circles removed: "+removed,...
    'FontSize',18);
saveas(gcf,'Large_circles_removed.png')

% Remove small circles that are more likely to be false positives
indices = find(abs(radii_beads)<3);
radii_beads(indices) = [];
centers_beads(indices,:) = [];
figure('Color','w'); % Figure 7
imshow(I);
c4 = viscircles(centers_beads,radii_beads,'Color','g');
title("Dynabeads detected: " + length(radii_beads),...
       'FontSize',18);
```

```
saveas(gcf,'small_circles_removed.png')

%% Detect and measure cells

[centers_cells,radii_cells] = imfindcircles(I3,[15 35],...
       'Sensitivity',0.9,...,
       'EdgeThreshold',0.2,...
       'Method','twostage');

%% Superimpose circular labels for detected cells and Dynabeads

figure('Color','w');
imshow(I); % Figure 8
% display cell detections
c5 = viscircles(centers_cells,radii_cells,'Color','b');
% display Dynabead detections
c6 = viscircles(centers_beads,radii_beads,'Color','r');
title("Detected cells (blue): "+length(radii_cells)+" | Detected Dynabeads
(red): "+length(radii_beads),...
       'FontSize',18);
saveas(gcf,'cells_and_beads1.png')

%% Find connected components in binary image

figure('Color','w'); % Figure 9
BW = imbinarize(I);
imshow(BW);
title('Binary', 'FontSize', 18);
saveas(gcf,'binary.png')

figure('Color','w'); % Figure 10
se = strel('disk', 8); % create a disk-shaped structuring element
BW2 = imdilate(BW, se);
imshow(BW2);
title('Dilated Binary Image', 'FontSize', 18);
saveas(gcf,'Dilated_binary.png')
% find connected components in the binary image
cc = bwconncomp(BW2);

%% Identify and extract the biggest cell-bead cluster

% Compute geometric measurements of the labeled objects using regionprops
stats = regionprops(cc, 'Area','PixelIdxList','PixelList');
area_values = [stats.Area];
% Obtain the index of the region with the largest area

[maxValue,idx] = max([stats.Area]);

% Obtain row and column coordinates of boundary pixels of the largest
% region, disregarding holes enclosed by connected component regions
boundaries = bwboundaries(BW2,'noholes');
x_region = boundaries{idx}(:, 2);
y_region = boundaries{idx}(:, 1);

% x- and y-coorindates of centroids of detected cells
x_cellCircles = centers_cells(:,1);
y_cellCircles = centers_cells(:,2);

% x- and y-coorindates of centroids of detected beads
x_beadCircles = centers_beads(:,1);
y_beadCircles = centers_beads(:,2);

%====== Find cells that lie within the biggest cell-bead cluster ====%
```

```
[in_cells,on_cells] =
inpolygon(x_cellCircles,y_cellCircles,x_region,y_region);
% number of cell centroids lying inside the tested region
numel(x_cellCircles(in_cells))
% number of cell centroids lying on the edge of the tested region
numel(x_cellCircles(on_cells))
% number of cell centroids lying outside of the tested region
numel(x_cellCircles(~in_cells))

%====== Find beads that lie within the bigges cell-bead cluster ====%
[in_beads,on_beads] =
inpolygon(x_beadCircles,y_beadCircles,x_region,y_region);
% number of bead centroids lying inside the tested region
numel(x_beadCircles(in_beads))
% number of bead centroids lying on the edge of the tested region
numel(x_beadCircles(on_beads))
% number of bead centroids lying outside of the tested region
numel(x_beadCircles(~in_beads))

%% Remove all labels that lie in the biggest cell-bead cluster

% Delete cells in cluster
centers_cells(in_cells,:) = [];
radii_cells(in_cells) = [];

% Delete beads in cluster
centers_beads(in_beads,:) = [];
radii_beads(in_beads) = [];

figure('Color','w'); % Figure 11
imshow(I);
c7 = viscircles(centers_cells,radii_cells,'Color','b');
c8 = viscircles(centers_beads,radii_beads,'Color','r');
title("[Final] Detected cells not in the huge cluster (blue):
"+length(radii_cells)+" | Detected Dynabeads (red): "+length(radii_beads),...
       'FontSize',18);
saveas(gcf,'cells_and_beads2.png')

figure('Color','w'); % Figure 12
plot(x_region,y_region); % boundary defined by the largest cell-bead cluster
axis equal
hold on
plot(x_cellCircles(in_cells),y_cellCircles(in_cells),'b+'); % cells inside
plot(x_cellCircles(~in_cells),y_cellCircles(~in_cells),'bo'); % cells outside
plot(x_beadCircles(in_beads),y_beadCircles(in_beads),'r+'); % beads inside
plot(x_beadCircles(~in_beads),y_beadCircles(~in_beads),'ro'); % beads outside
hold off
set(gca, 'ydir', 'reverse') % flip the y-axis so that the output plot mirrors
the original image
title('"+" = inside of cluster | "o" = outside of cluster | blue = cells |
red = Dynabeads', 'FontSize', 18);
saveas(gcf,'cells_and_beads_inoutofcluster.png')

%% Find cell-bead clusters

num_cells = length(centers_cells); % numebr of cells
num_beads = length(centers_beads); % numebr of beads
tol = 40; % set the overlap tolerance
cell_bead_contacts = zeros(num_cells,1);

for i = 1:num_cells
       for j = 1:num_beads
              d_ij = sqrt((centers_cells(i,1)-
```

```
centers_beads(j,1)).^2+(centers_cells(i,2)-centers_beads(j,2)).^2);
       k = radii_cells(i)+radii_beads(j)+tol;
       if d_ij < k % if the cell and the bead are in contact
              cell_bead_contacts(i) = cell_bead_contacts(i)+1; % save the
detected contacts
              end
       end
end

contacts_idx = find(cell_bead_contacts);

figure('Color','w'); % Figure 13
plot(x_cellCircles(contacts_idx),y_cellCircles(contacts_idx),'g+',...
       'MarkerSize',12,...
       'LineWidth',2);
axis equal
hold on
plot(x_cellCircles(in_cells),y_cellCircles(in_cells),'b+') % cells inside
plot(x_cellCircles(~in_cells),y_cellCircles(~in_cells),'bo') % cells outside
plot(x_beadCircles(in_beads),y_beadCircles(in_beads),'r+') % beads inside
plot(x_beadCircles(~in_beads),y_beadCircles(~in_beads),'ro') % beads outside
hold off
set(gca, 'ydir', 'reverse')
title('Contacts','FontSize', 18);
saveas(gcf,'contacts_inoutofcluster.png')

% Update x and y coordinates of cells and beads to include only those
% outside of the cluster
x_cellCircles = centers_cells(:,1);
y_cellCircles = centers_cells(:,2);
x_beadCircles = centers_beads(:,1);
y_beadCircles = centers_beads(:,2);

figure('Color','w'); % Figure 14
imshow(I);
hold on

plot(x_cellCircles(contacts_idx),y_cellCircles(contacts_idx),'g+',...
'MarkerSize',12,...
'LineWidth',2);
title('Cell-bead contacts','FontSize', 18);
saveas(gcf,'contacts_superimposed.png')

figure('Color','w'); % Figure 15
h = histogram(cell_bead_contacts)
xlabel('Number of detected Dynabead
contacts','FontSize',14,'FontWeight','bold')
ylabel('Number of detected cells','FontSize',14,'FontWeight','bold')
ax = gca;
ax.XAxis.FontSize = 12;
ax.YAxis.FontSize = 12;
saveas(gcf,'histogram.png')

num_boundBeads = length(find(bead_cell_contacts));
num_unboundBeads = num_beads - num_boundBeads;

figure('Color','w'); % Figure 16
pie_chart = [num_unboundBeads num_boundBeads];
p = pie(pie_chart);
labels = {'Unbound beads','Cell-bound beads'};
legend(labels,'FontSize',14)
set(findobj(p,'type','text'),'fontsize',14
```

FLUORESCENCE IMAGING (HYDROGEL SIDE VIEW)

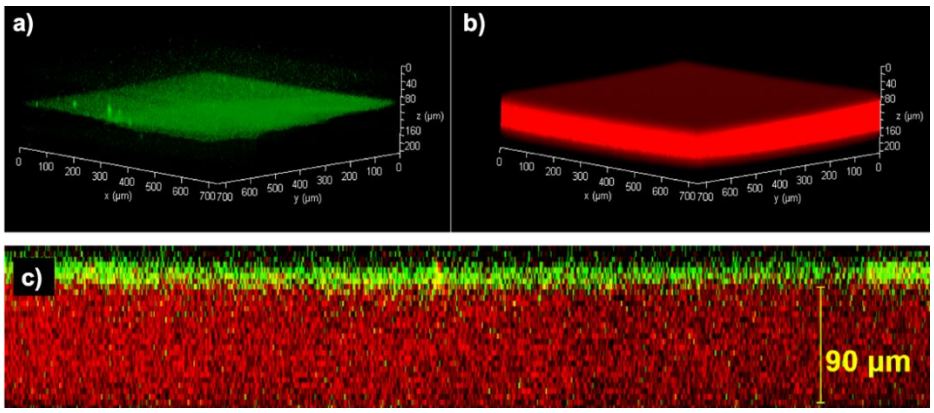

Figure S3. Representative fluorescence images (3D reconstruction from z-stacks) of an anti-CD3-coated PA hydrogel with (**a**) the anti-CD3 layer labelled with FITC-conjugated secondary antibody (green) and (**b**) the bulk of the gel labelled with Alexa Fluor 568 (red). The cross-sectional merged view of (**a**) and (**b**) is shown in (**c**).

POST-STIMULATION (48 HOURS) CELL VIABILITIES

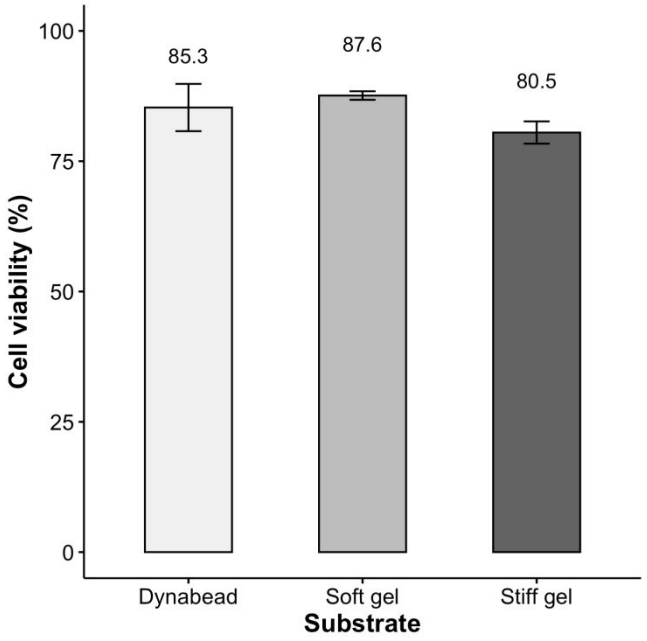

Figure S4. Viabilities of Jurkat T cells after 48 hours of stimulation by different substrate materials. All substrates presented anti-CD3/CD28 as stimulatory cues. Soft gel refers to 7.1 ± 0.4 kPa. Stiff gel refers to mean stiffness of 50.6 ± 15.1 kPa. Data = mean ± standard deviation. Values displayed above bars indicate mean cell viabilities. N = 3. Viability was measured using the NucleoCounter® NC-200™ automated cell counter running the Viability and Cell Count Assay.