SUPPLEMENTARY APPENDIX

# Megakaryocyte volume modulates bone marrow niche properties and cell migration dynamics

**Maximilian G. Gorelashvili,[1*] Oğuzhan Angay,[2*] Katherina Hemmen,[2] Vanessa Klaus,[1] David Stegner[1,2] and Katrin G. Heinze[2]**

[1]Institute of Experimental Biomedicine, University Hospital Würzburg and [2]Rudolf Virchow Center for Experimental Biomedicine, University of Würzburg, Würzburg, Germany

**MGG and OA contributed equally to this work*

 doi:10.3324/haematol.2018.202010

Received: July 15, 2018.

Accepted: June 25, 2019.

Pre-published: June 27, 2019.

Correspondence: KATRIN HEINZE - katrin.heinze@virchow.uni-wuerzburg.de

DAVID STEGNER - stegner@virchow.uni-wuerzburg.de

# Megakaryocyte volume modulates bone marrow niche properties and cell migration dynamics

Maximilian G. Gorelashvili[1], Oğuzhan Angay[2], Katherina Hemmen[2], Vanessa Klaus[1], David Stegner[1], Katrin G. Heinze[2]

**Author affiliations:**

[1]Institute of Experimental Biomedicine, University Hospital Würzburg, Würzburg, Germany

[2]Rudolf Virchow Center for Experimental Biomedicine, University of Würzburg, Würzburg, Germany

MGG and OA contributed equally to this work.

**Correspondence:** katrin.heinze@virchow.uni-wuerzburg.de, stegner@virchow.uni-wuerzburg.de

## SUPPLEMENTAL METHODS

### Two-photon microscopy image analysis

To obtain the time-dependent mean-square displacement, *MSD*(*t*), of the neutrophils in the bone marrow, the time-series z-stacks needed to be analyzed in several steps: image pre-processing, segmentation and tracking, calculation and analysis of MSD-curves.

*Image pre-processing*. Due to breathing of the mice, slight, periodic spatial shifts between the time-series images occurred. These were corrected by the Huygens Object stabilizer (Scientific Volume Imaging, Netherlands). We selected stabilization over *t* of the vessel channel via time cross-correlation with no rotation detection and exported the full range. Next, the images were smoothed and sharpened with FIJI (36). Here, we first applied a 2D median filter with a radius of one pixel and then an unsharp mask with a width of ten pixel and a weight of 0.4.

*Segmentation and tracking*. Segmentation and tracking of the neutrophils was performed with Imaris 9.2.1 (Bitplane AG, Switzerland). The neutrophils were selected using the "spots wizard" as a rather uniform, ellipsoidal size and shape was expected. We used the whole dataset as region of interest and enabled "track over time". The estimated lateral diameter of the neutrophils was estimated to 6 µm and the axial length to 18 µm. With background subtraction enabled, we set the quality threshold for selection of neutrophils to 15 %. Next, the neutrophils were tracked over time using the autoregressive motion algorithm with a maximal distance of 8 µm per step, a maximal gap of three frames and fill gaps enabled. In a final step, we filtered the tracks for their duration ($n$*$\Delta t$) and included only those tracks longer than 15 min in our further analysis. In average, 124 neutrophils were tracked per measurement (137 in control mice, 99 in depleted mice, minimum 62).

For further analysis, the instantaneous velocity, the squared displacement and the time since track start were exported from Imaris 9.2.1.

*Calculation and analysis of MSD-curves*. The $MSD(t)$ of the $N$ neutrophils was calculated from the exported squared displacement $d^2(p_i, p_{i+n})$ and the time since track start, $n$*$\Delta t$:

$$MSD(t) = \frac{1}{N-n}\sum_{i=1}^{N=n} d^2(p_i, p_{i+n}) \qquad \text{eq. 1}$$

The obtained curves were fitted using OriginPro 2018 with an asymptotic model to determine the limit $L$:

$$MSD(t) = L - R * r^t \qquad \text{eq. 2}$$

where $R$ is the response range and $r$ the response rate. Note that neither $R$ nor $r$ have a physically sensible meaning.

Additionally, the first ~ 25 % of each $MSD(t)$ curve (39) – corresponding to 42 min of the simulations and 9 min of the experimental results – were fitted with a 3D

anomalous diffusion model to obtain an apparent diffusion coefficient $D_{app}$ and the diffusion exponent $\alpha$:

$$MSD(t) = 8D_{app}t^{\alpha} \qquad \text{eq. 3}$$

**Tissue preparation for LSFM**

Tissue preparation was performed as previously described (15). Briefly, mouse MKs/platelets and vasculature were labeled by intravenous injection of anti-GPIX Alexa Fluor 750 derivative ((15), 0.6 µg/g body weight) and anti-CD105 Alexa Fluor 647 (BioLegend, San Diego, CA, USA, 0.4 µg/g body weight), respectively. Animals were anesthetized according to the local regulations and transcardially perfused with PBS and 4% paraformaldehyde (Sigma-Aldrich, Schnelldorf, Germany). Intact bones were harvested, postfixed, decalcified, dehydrated and cleared according as described previously (15).

**Light sheet fluorescence microscopy**

Cleared bones were imaged using a home-built light-sheet fluorescence microscope as described previously (15). For image acquisition three different detection channels (green for bone and bone marrow, red for vasculature, far-red for MKs) with three corresponding excitation wavelengths (491, 642, and 730 nm) were used: Images were saved as tiff files with a voxel size of 0.5 x 0.5x2 $\mu m^3$.

**LSFM: MK image processing**

In this study we focus on optimization of MK segmentation; other segmentation such as of blood vessels, bones, and vessel interspace measurements were performed as described in (15).

Shared preprocessing

The four different analysis pipelines (I-IV), shared the same image preprocessing steps performed in FIJI (36) and Imaris® 8.4 (Bitplane AG, Zurich, Switzerland) in the following order:

1. **FIJI:** After an initial noise removal step (median filter, 2 px radius), crosstalk from the blood vessel channel into MK channel was removed. The following FIJI functions were used for this step: ROI tool, ROI manager, image calculator. In each image stack 10 regions of interests (ROIs) containing vessel crosstalk were

outlined (polygon). In both channels avg. intensities were measured and the ratio calculated for each ROI separately. The mean of all 10 ROIs was used for following calculation:
$MK_{decrossed}$= MK CH - (mean Crosstalk ratio * Vessel CH)

2. **FIJI:** Cell fragments and small staining artifacts (speckles) were removed in $MK_{decrossed}$ and Vessel channel:
   The respective channel was duplicated and binarized (initial Otsu auto-threshold with manual adjustment). All non-artefacts were filtered out with the Particle remover plugin (https://imagej.nih.gov/ij/plugins/particle-remover.html) using following parameters: Vessel channel: size (pixel^2) 81-infinity; circularity 0-0.4
   $MK_{decrossed}$: size (pixel^2) 100-infinity; circularity 0-0.4
   The filtered binary files were converted back to the original 16 bits file format, squared (process->math->square) and subtracted from the originals.
3. **Imaris:** All channels were normalized (normalize layers) and converted to 32 bits.
4. **FIJI:** Local contrast was enhanced by using a large radius unsharp mask (radius 50 px, weight 0.8). The output was median filtered (2 px).
5. **Imaris:** The channels were background subtracted before further processing. The filter diameter was set slightly larger (+ 5µm) than the objects of interest: MK channel 40 µm; Vessel channel 40-55 µm. The resulting MK channel is referred as $MK_{noseed}$ in the following.

Segmentation pipelines

*Pipeline I (membrane algorithm)*

This pipeline was performed exclusively with the Imaris Cell (membrane) algorithm:

$MK_{noseed}$ was used for this pipeline and intensity thresholds were manually determined. They were set to low values for achieving any segmentation results (all samples @300 compared to 700-900 depending on data set). Other parameters were defined as follows: Minimum diameter 10 µm, membrane detail 1 µm, quality 50% of auto value, minimum volume 525 µm³, maximum volume 65000 µm³. Cell objects were exported as iso-surfaces for further use and conversion artifacts below 525 µm³ were deleted.

*Pipelines II-IV segmentation*

The Imaris cell (soma) algorithm was used for pipelines II-IV. Pipeline II was performed in one main step, while pipeline III consisted of two main steps, both applied to $MK_{noseed}$. Pipeline IV shared the same general steps as III but was extended with an additional preparational step by creating artificial MK somata / seeding points (referred to as $MK_{cleaned\_seed}$). For all pipelines: Segmented cell objects were exported to iso-surfaces for later use and statistical comparison, since only iso-surfaces allow masking operations in Imaris, a crucial part of pipelines III and IV.

*Pipeline II: one-pass*

Imaris cell algorithm with the following parameters was used for pipeline II: Cell (soma) algorithm, smoothing 2µm, intensity threshold initially set to 60% of auto

value and manually fine-tuned (histogram-based), seed point diameter 10 µm, quality 50% of auto value, filter: minimum volume 525 µm³. Cell objects were exported as iso-surfaces for further use and conversion artifacts below 525 µm³ were deleted.

*Pipeline III: two-pass*

Pipeline III was performed in two consecutive steps using the Imaris cell (soma) algorithm. First segmenting the larger cells and in the second step the smaller MKs.

**Step I** (MK20) was applied on $MK_{noseed}$ using following parameters: Imaris cell (soma) algorithm, smoothing 2µm, intensity threshold first set to 60% of auto value and manually fine-tuned (histogram-based), seed point diameter 20µm, quality 50% of auto value, filter: volume 4188-45000 µm³. Cell objects were exported as iso-surfaces for further use and conversion artifacts below 525 µm³ were deleted.

In **step II** (MK10) $MK_{noseed}$ was masked with the iso-surfaces from **step I** (duplicate CH, set inside=0, untick outside). The resulting channel was used to create the remaining MKs using Imaris cell soma algorithm with adjusted parameters: smoothing 1 µm, manual threshold setting above the value of step II (aided by the histogram), seed point diameter 10 µm, quality 50% of auto value, filter: minimum volume 525 µm³. Cell objects were exported as iso-surfaces and conversion artifacts below 525 µm³ were deleted.

The iso-surfaces populations MK20 and MK10 were merged in a final step for statistics export.

*Pipeline IV*

In contrast to II and III, pipeline IV contained additional processing steps for the creation of artificial MK somata / seeding cores prior to segmentation:

1. **FIJI:** The MK channel ($MK_{noseed}$) was duplicated and auto-thresholded (otsu, dark background). The resulting binary ($MK_{plain}$) was duplicated again ($MK_{core}$). Close (3 iterations) and fill holes binary function was applied on $MK_{core}$. $MK_{plain}$ was subtracted from $MK_{core}$. FIJI particle remover plugin was applied (size 0-100 px) on the result and datatype was set back to 32 bit ($MK_{finalcores}$).
2. **Imaris:** $MK_{finalcores}$ was used to create iso-surfaces (no smoothing, auto intensity threshold, filters: BoundingBox AAZ 5 µm, sphericity 0.65-1). The iso-surfaced were used to mask (inside 6000 / outside unticked) the preprocessed MK channel ($MK_{noseed}$), resulting in new a MK channel with artificial MK somata /seeding cores ($MK_{seed}$).
3. **Imaris:** A final artifacts removal step was performed. Iso-surfaces were created with $MK_{seed}$, using following parameters: smoothing 1 µm, no splitting, intensity threshold manually set to ~25% of the auto-threshold, filters: maximum volume 524 µm³, sphericity 0.75-1. $MK_{seed}$ was masked with resulting iso-surfaces (inside surface 0, outside unticked, duplicate channel) creating a new channel $MK_{cleaned\_seed}$.
4. **Imaris:** MKs were segmented by applying same two sequential steps with the same parameters of pipeline III on $MK_{cleaned\_seed}$.

## Virtual slice generation

Imaris was used to create virtual slices. In total, 10 virtual slices per image stack were created, with a slice thickness of 10 µm and an interslice distance of 50 µm. We divided them into two interleaving groups, 5 odd and 5 even numbered, and performed separate distance transformation map measurements to avoid readout artefacts.

For better comparison, the same pipeline IV datasets were used. We created ground truth data by masking MK and Vessel iso-surfaces into the virtual slices.

First, the vessels were masked (binary, inside 5000 / outside 0) into the slices. Then they were re-segmented (iso-surfaces, auto threshold, 1µm smoothing, minimum volume 10 voxels), and distance transformation maps were created.

The vessel interspace measurement was performed as described in Stegner *et al*. (15).

MK iso-surfaces were masked as virtual slices into $MK_{cleaned\_seed}$ channel (inside unticked / outside 0). Then, the main two segmentation steps of pipeline III / IV were applied on the slices. At step 2 the minimum volume was reduced to 10 voxels.

## Computational simulations

Static simulation of megakaryocyte distribution in the bone marrow

The algorithm for the simulation of megakaryocyte distribution in the bone was originally developed in (15) to model the distribution of real-shaped MKs and vessel-system derived from experimental light-sheet fluorescence microscopy images. After segmentation in Imaris the parameters of the identified objects like size, volume, shape or position were exported. This information was used to calculate the ratio of volume MK to volume bone marrow (normalized MK number is calculated as number of MKs per $mm^3$ of the sample and the MK volume is directly determined during segmentation). These objects were also used for the computational simulations. The algorithm is depicted in **Fig. S4** and the used Matlab-scripts are given below.

Here, we extended this simulation by implementation of ideal-shaped MKs and an artificial, regular shaped vessel system.

In short, a Matlab-based algorithm was developed, allowing to predefine different cell shapes and a stable template into which cells are randomly placed. As we used image-derived objects, consequently, the algorithm is voxel-based and all objects, such as vasculature or cells, are represented by 3D tensors. Furthermore, the algorithm performs sequential addition of MKs into the simulation space. The algorithm has no restrictions concerning the size and shape of the template or the cells. The quality of the simulation is limited only by the spatial resolution of the used image stacks, which can be improved either by increased imaging resolution or interpolation of voxel values. Instead of using binary images of the vasculature, a distance transformation map (DTM) of the channel is used as a template, allowing direct calculation of cell-to-vessel distances. Each voxel value V represents the distance to the next vascular neighbor: $V < 0$ for vessel lumen, $V = 0$ for vessel wall, $V > 0$ for intravascular space. The vasculature DTM of size $d_1$ x $d_2$ x $d_3$ is converted into a tensor $D$ of type ($d_1$, $d_2$, $d_3$) where $D_{ijk}$ corresponds to the voxel values of the

DTM. In addition, a second tensor *B* of type ($d_1$, $d_2$, $d_3$) is introduced, which represents a binary image of the vasculature:

$$B_{ijk} = \begin{cases} 1, if\ D_{ijk} \leq 0 \\ 0, if\ D_{ijk} > 0 \end{cases} \qquad \text{eq. 4}$$

The algorithm is able to include different cells into the simulation, enabling to predefine the size and sphericity distribution of simulated MKs. All *m* binary MK image stacks are transformed into 3D tensors $C_p$ of individual sizes ($p_1$ + 1, $p_2$ + 1, $p_3$ + 1), where $C_{p,ijk}$ = 1 represents voxels belonging to a cell and values outside of the cell are set to empty "not a number *(NaN*)". Before the actual simulation, the number of simulated MKs *n* and the sequence of used MK tensors are predefined. In addition, a tensor *M* of size ($d_1$, $d_2$, $d_3$) is generated, where at the beginning of the simulation all values are set to 0 and are changed when a MK is added to the space. For random addition of a new $m^{th}$ cell into the bone marrow space, first, a point outside the vasculature with $B_{ijk}$ = 0 is randomly chosen and the cell is placed. Consequently, the cell with the tensor $C_p$ will occupy the space ([*i, i* + $p_1$], [*j, j* + $p_2$], [*k, k* + $p_3$]). If $i + p_1 > d_1$, $j + p_2 > d_2$ or $k + p_3 > d_3$ the cell is partially located outside the template and a new position for the cell is chosen.

Next, the overlap of the $m^{th}$ cell with the vasculature or other MKs is calculated:

$$O^{VC} = \sum_{\tilde{i}\tilde{j}\tilde{k}} O^{VC}_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} = \sum_{\tilde{i}\tilde{j}\tilde{k}} B_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} C_{\tilde{p}_1, \tilde{p}_2, \tilde{p}_3} \qquad \text{eq. 5a}$$

$$O^{MC} = \sum_{\tilde{i}\tilde{j}\tilde{k}} O^{MC}_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} = \sum_{\tilde{i}\tilde{j}\tilde{k}} \sigma(M_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} C_{\tilde{p}_1, \tilde{p}_2, \tilde{p}_3}) \qquad \text{eq. 5b}$$

where $$\sigma(x) = \begin{cases} 1, if\ x > 0 \\ 0, if\ x = NaN \end{cases}$$

and $\tilde{p}_1, \tilde{p}_2, \tilde{p}_3 \in \mathbb{N}$, $\tilde{p}_1 \in [1, p_1]$, $\tilde{p}_2 \in [1, p_2]$, $\tilde{p}_3 \in [1, p_3]$. If the overlaps $O^{VC}$ and $O^{MC}$ exceed predefined values, a new position for the cell is chosen. Otherwise MK tensor *M* is modified to *M'* as follows:

$$M'_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} = m * M_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3} \qquad \text{eq. 6}$$

Such a modification of the MK tensor results in MKs distinguishable by unique voxel values and thus facilitates cell segmentation for visualization purpose. In addition, cell-to-vessel distance is calculated as the infimum value of $B_{i+\tilde{p}_1, j+\tilde{p}_2, k+\tilde{p}_3}$. The last step of the algorithm stores cell-to-vessel distances and the tensor *M*.

Dynamic simulation of cell migration in the bone marrow

The cell migration algorithm is based on the computation modeling program described above and uses *real microscopy-derived 3D images* of bone marrow vasculature and MKs as templates (**Fig. S5**) and during migration single steps are simulated assuming a fixed step duration. The algorithm is depicted in **Fig. S6** and the used Matlab-scripts are given below.

First, the template is loaded and vessel and MK space tensors are generated to later enable cell-vasculature and cell-MK overlap calculations. Subsequently, the 3D image of the migrating cell is loaded and a corresponding 3D tensor is built. The starting position $\vec{P}(t = 0)$ of the migrating cell is randomly generated according to the algorithm for MK distribution simulation. Next, a random $i^{th}$ step vector $\vec{S}(t = i)$ is

generated depending on different parameters. The step is performed in 3D, where in each direction discrete values of a normal distribution function of predefined mean value and standard deviation are taken as basic instantaneous velocity. In addition to the basic velocity, chemotaxis effects can be added. The source of the attractive or repulsive chemotactic signal can be any predefined template, which in this special case was chosen to be the vasculature. The signal displays a linear gradient and thus correlates with the distance transformation map of the template. The signal is detected by cell membrane voxels and the net attraction is calculated for all three dimensions. The net attraction is then multiplied with a predefined chemotaxis parameter *C* to calculate additional step size. This approach reflects the physiological chemotactic sensing mechanisms of different types of cells. To simulate cell migration without chemotaxis, the parameter *C* was set to the value 0. At every step the cell is moved to the new position $\vec{P}(t=i) = \vec{P}(t=i-1) + \vec{S}(t=i)$ which is calculated by the addition of the previous position and the step vector. If the migrating cell leaves the template or has an overlap with a MK, the cell is moved back to the previous position with $\vec{P}(t=i) = \vec{P}(t=i-1)$. This step enables to mimic the phase of no movement, which is needed for cytoskeleton reorganization due to change of the direction of migration. If the cell does not collide with a MK or leave the template, its overlap with the vasculature is calculated. Here, the algorithm gives the possibility to simulate the process of a cell entering the blood flow by introducing a predefined probability. In case of intravasation, the simulation will be stopped and all following positions will be stored as $\vec{P}(t \geq i) = \vec{P}(t=i)$. Otherwise, $\vec{P}(t=i)$ is stored and a new step is simulated. After a predefined number *n* of positions, the simulation is stopped and all data are saved.

From the stored positions, the mean-squared displacement (*MSD*) as a function of stepsize Δ*t* was calculated:

$$MSD(\Delta t) = \frac{1}{N-n}\sum_{i=1}^{N=n} d^2(\vec{P}_i, \vec{P}_{i+n}) \qquad \text{eq. 7}$$

where $d^2(\vec{P}_i, \vec{P}_{i+n})$ is the squared distance between positions.

Each simulation was run 200 – 600 times. HSCs and neutrophils had a diameter of 6 µm and 10 µm, respectively, with a sphericity of 1 based on published data (37, 38).

In our simulations we varied systematically the vessel stickiness (prob-EV: 100% or 50 %), the chemotaxis parameter (*C*: 0.4, 0.2 0.1 or 0) and the velocity of the cells ($v^{low}$: 2 ± 1 µm, $v^{high}$: 3 ± 1 µm). all simulations were performed in the absence and presence of static MKs.

Matlab scripts and Ilastik training dataset were uploaded at Zenodo under doi: 10.5281/zenodo.3144732.

## ALGORITHM CODES

### Original algorithm code for static simulation of spatial megakaryocyte distribution

The simulations were performed by defining the needed parameters and respective folders, including the template image stacks, in the matlab file simulate.m. simulate.m was built in a modular way and uses further algorithms indicated in orange color, the codes of which are presented below. All entries showed in blue represent examples. All entries in green are annotations of the code.
The algorithm is visualized in **Fig. S4**.

**isborder.m**

```
function [x]=isborder(y,z)
if y<z+1 & y>0
x=y;
else x=0;
end
```

**Simulate.m**

```
for m=1:10 %number of planned simulations
ncell=700; %number of simulated cells; ncell must be 0 modulo number of
different cell shapes;
numcelltemplate=20; %number of used cell templates; numbercellshape must be
0 modulo numbercelltemplate
cellfname1=['G\MKSimulation\MKmigration_old\Simulation_middle\cellpool\merg
edcellpool\']; %folder containing tiff image sequences of cell templates
folderpath1=['G:\MKSimulation\MKmigration\StaticSimul\Simulations\']; %main
folder for saving simulation results
folderpath2=['simulation_20170517_artV_realMK']; %subfolder for saving
simulation results
folderpath3=['' num2str(m) ''];
folderpath=[folderpath1 folderpath2 folderpath3];
mkdir(folderpath);
numbercellshape=20; %number of different cell shapes which normally equals
numbercelltemplate
aXY=0.5; %voxel size in X and Y directions
aZ=2; %voxel size in Z direction
savesimulationdata=0; %if value 1 tiff image sequence of simulation is
saved
MKadjusted=3;
dtfname =
'G:\MKSimulation\MKmigration\StaticSimul\Templates\DistanceCrop.tif';
loadfiles
randomcells_overlap
clear
end
```

**loadfiles.m**

```
%-----------------------------------------
% load the distance transformation map stack
%-----------------------------------------
dtstack0=tiffread2(dtfname);
for k = 1:length(dtstack0)
dtstack1=dtstack0(k);
```

```
DTStack(:,:,k) = dtstack1.data;
end
ImStack=DTStack<=0; % 1 inside the vessel, 0 outside the vessel
[x,y,z]=ind2sub(size(DTStack), find(DTStack>0)); %coordinates of voxels
outside the vessels
D=[x,y,z]; %coordinates of voxels inside the vessel; will be changed during
cell generation->voxels outside the vessels - cells outside the vessels
clear x y z dtstack0 dtstack1 k
%%
[x,y,z]=ind2sub(size(DTStack), find(DTStack<0));
C=[x,y,z]; %coordinates of voxels inside the vessels, needed for cell at
vessels
generation
clear x y z
lenX=length(DTStack(1,:,1));
lenY=length(DTStack(:,1,1));
lenZ=length(DTStack(1,1,:));
ImX=lenX; ImY=lenY; ImZ=lenZ;
ImStackMov=zeros(lenY, lenX, lenZ) ;
%%
%------------------------
% load the cell stacks
%------------------------
cellStack=[];
lencX=[];
lencY=[];
lencZ=[];
for i=1:numcelltemplate %number of different cells in the pool
cellfname=[cellfname1 'cell' num2str(i+10) '.tif'];
infocell = imfinfo(cellfname);
cellStack1 = [];
numberOfImages = length(infocell);
for k = 1:numberOfImages
currentImage = imread(cellfname, k, 'Info', infocell);
cellStack1(:,:,k) = currentImage;
cellStack1(:,:,k) = cellStack1(:,:,k)/max(max(cellStack1(:,:,k)));
end
[x,y,z]=ind2sub(size(cellStack1), find(cellStack1)); %positions of 1 pixels
of cell stack
eval(['B' num2str(10+i) '(:,:,:)=[x,y,z];'])
eval(['cellStack' num2str(10+i) '(:,:,:)=cellStack1;'])
lencX(i)=length(cellStack1(1,:,1));
lencY(i)=length(cellStack1(:,1,1));
lencZ(i)=length(cellStack1(1,1,:));
end
clear k currentImage info cellfname infocell x y z cellStack1
numberOfImages
currentImage cellStack i
```

**Randomcells_overlap.m**

```
%%
%=======================================
% Create cells in a random way
%=======================================
% define the shape of each cell
randhelp3=[1:20];
randhelp1=[9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9]; %Volume distribution
adjusted
```

```
randhelp2=[6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6]; % Diameter
distribution adjusted
numberofblocks=ncell/numbercellshape;
sa=[];
for i=1:numberofblocks
if MKadjusted==1
sa=[sa, randhelp1(randperm(20))];
elseif MKadjusted==2
sa=[sa, randhelp2(randperm(20))];
else
sa=[sa, randhelp3(randperm(20))];
end
end
%%
%----------------------------------------
% start: create a point outside the vessels
%----------------------------------------
disves=nan(ncell,1);
cellstartpos0=nan(3,ncell);
for i=1:ncell
i
tic
helpstartpos1=10^7; %for vessel-cell interaction; the number is here higher
than any cell volume; enables starting the while-loop
helpstartpos2=10^7; %for cell-cell interaction; the number is here higher
than any cell volume; enables starting the while-loop
Cint=0; %Voxel volume of the cell, which is calculated later; value 0
enables starting the while-loop
eval(['B=B' num2str(sa(i)+10) ';']);
eval(['cellStack=cellStack' num2str(10+sa(i)) ';']);
while helpstartpos1>0 || helpstartpos2>Cint*0.03 %helpstartpos=0 => start
position is found
try
cellstartpos0(:,i)= D(randi([1 length(D)]),:); % creates a random point in
the Vessel space
%to avoid cells outside the vessel space:
minx=cellstartpos0(1,i);
maxx=isborder(cellstartpos0(1,i)+lencX(sa(i))-1,ImX);
miny=cellstartpos0(2,i);
maxy=isborder(cellstartpos0(2,i)+lencY(sa(i))-1,ImY);
minz=cellstartpos0(3,i);
maxz=isborder(cellstartpos0(3,i)+lencZ(sa(i))-1,ImZ);
if maxx==0 || maxy==0 || maxz==0 %=>cell is partially outside the stack
helpstartpos1=10^7; %for vessel-cell interaction; the number is here higher
than any cell volume
helpstartpos2=10^7; %for cell-cell interaction
else
help1=DTStack(miny:maxy,minx:maxx,minz:maxz); %part of the Vessel space
colocalizing with the cell at t=0
[x,y,z]=ind2sub(size(help1), find(help1<=0)); %finds the values larger 0 in
help1
help2=[x,y,z]; %coordinates of values larger 0 in help1
help3=ImStackMov(miny:maxy,minx:maxx,minz:maxz); %part of the cell stack
colocalizing with the cell at t=0
[x,y,z]=ind2sub(size(help3), find(help3));
help4=[x,y,z]; %coordinates of values larger 0 in help3
clear x y z
helpstartpos1= sum(ismember(B(:,:,:),help2,'rows')); %checks if vessel and
cell colocalize
```

```
helpstartpos2= sum(ismember(B(:,:,:),help4,'rows')); %checks if cell and
cell colocalize
Cint=sum(sum(sum(cellStack))); %
end
catch
disp('error')
end
end
ImStackMov(miny:maxy,minx:maxx,minz:maxz)=ImStackMov(miny:maxy,minx:maxx,mi
nz:maxz
)+cellStack(:,:,:).*10*i;
%build disves
cellStack1=cellStack;
cellStack1(cellStack1==0)=NaN;
helpDTStack=DTStack(miny:maxy,minx:maxx,minz:maxz).*cellStack1 ;
disves(i,1)=min(min(min(helpDTStack)));
%change D
ImStack(miny:maxy,minx:maxx,minz:maxz)=ImStack(miny:maxy,minx:maxx,minz:max
z)+ImSt
ackMov(miny:maxy,minx:maxx,minz:maxz);
[x,y,z]=ind2sub(size(ImStack), find(ImStack==0)); %positions of 1 pixels of
vessel stack
D=[x,y,z];
toc
end
%%
%-------------------------
%save the images if needed
%-------------------------
if savesimulationdata==1
for j=1:lenZ
outputFileName = [folderpath '\image_T1001_Z' , num2str(1000+j) ,'.tif']
imwrite(ImStackMov(:, :, j), outputFileName, 'WriteMode',
'append','Compression','none');
end
end
save([folderpath '\simulation1.mat'])
```

### Original algorithm code for static simulation of spatial cell distribution

The simulation was performed by defining the needed parameters and respective folders, where the template images were stored, in the matlab file simulate.m. simulate.m was built in a modular way and uses further algorithms indicated in orange color, the codes of which are presented below. All entries showed in blue represent examples. All entries in green are annotations of the code.
The algorithm is visualized in **Fig. S6.**

**Simulate.m**

```
folderpath1=['G:\MKSimulation\MKmigration\MigrationSimul\Simulations\201705
31ctrl_3h\']; %main folder for saving the simulation results
folderpath2=['simulation_20170531_migr'];%subfolder for saving simulation
results
cellfname=['G:\MKSimulation\MKmigration\MigrationSimul\Templates\HSC.tif'];
%tiff containing the artificial HSC image
dtfname =
'G:\MKSimulation\MKmigration\MigrationSimul\Templates\DTMKvessel.tif';
```

```
%tiff containing the DTM of MK+Vasculature
dtfname1 =
'G:\MKSimulation\MKmigration\MigrationSimul\Templates\DTvessel.tif'; %tiff
containing the DTM of Vasculature
stepnumber=30000; %number of simulated steps
loadfiles
for m=1:200
folderpath3=['' num2str(m) ''];
folderpath=[folderpath1 folderpath2 folderpath3];
mkdir(folderpath);
p_walk=0.5; %probability to keep walking after vessel contact
meanvel=3; %mean migration velocity in [µm/10 min]
stdvel=2; %standard deviation of migration velocity in [µm/10 min]
chemotaxis=1; %1 for chemotaxis, 0 for random walk
chemf=0.1; %parameter for chemotaxis strength
neededparameters
randomcell
Chemotaxis2
clear ans cellcat cellpos cellstartpos0 disves disvesMKves gradx grady
gradz help1 help2 help3 helpDTStack1 helpstartpos1 i m maxx maxy maxz minx
miny minz stepx stepy stepz
end
```

**loadfiles.m**

```
%-------------------------------------------------------------------
% load the distance transformation map stack for vessel and cells
%-------------------------------------------------------------------
dtstack0=tiffread2(dtfname);
for k = 1:length(dtstack0)
dtstack1=dtstack0(k);
DTStack(:,:,k) = dtstack1.data;
end
[x,y,z]=ind2sub(size(DTStack), find(DTStack>0)); %coordinates of voxels
outside the vessels
D=[x,y,z]; %coordinates of voxels outside the vesselMK
clear x y z dtstack0 dtstack1 k
lenX=length(DTStack(1,:,1));
lenY=length(DTStack(:,1,1));
lenZ=length(DTStack(1,1,:));
ImX=lenX; ImY=lenY; ImZ=lenZ;
%%
%-------------------------------------------------------------------
% load the distance transformation map stack for vessels only
%-------------------------------------------------------------------
dtstack0=tiffread2(dtfname1);
for k = 1:length(dtstack0)
dtstack1=dtstack0(k);
DTStack1(:,:,k) = dtstack1.data;
end
clear x y z dtstack0 dtstack1 k
%%
%------------------------
% load the cell stacks
%------------------------
lencX=[];
lencY=[];
lencZ=[];
infocell = imfinfo(cellfname);
```

```
cellStack1 = [];
numberOfImages = length(infocell);
for k = 1:numberOfImages
currentImage = imread(cellfname, k, 'Info', infocell);
cellStack1(:,:,k) = currentImage;
cellStack1(:,:,k) = cellStack1(:,:,k)/max(max(cellStack1(:,:,k)));
end
[x,y,z]=ind2sub(size(cellStack1), find(cellStack1)); %positions of 1 pixels
of cell stack
B11(:,:,:)=[x,y,z];
cellStack11(:,:,:)=cellStack1;
lencX=length(cellStack1(1,:,1));
lencY=length(cellStack1(:,1,1));
lencZ=length(cellStack1(1,1,:));
clear k currentImage info cellfname infocell x y z cellStack1
numberOfImages currentImage cellStack i
```

**Neededparameters.m**

```
%%
%==================================
% needed parameters
%==================================
% µm/pixel in X-Y plane
aXY=0.5;
% µm/pixel in Z plane
aZ=2;
%velocity in pixel/10min
meanvelMKxy=meanvel/aXY;
%velocity in pixel/10min
meanvelMKz=meanvel/aZ;
%velocity standard deviation in pixel/5min
stdvelMK=stdvel/aXY;
stdvelMKz=stdvel/aZ;
%experiment time, where one step corresponds to 10 min
t=[1:stepnumber];
```

**Randomcell.m**

```
%%
B=B11;
cellStack=cellStack11;
%----------------------------------------
% start: create a point outside the vessels
%----------------------------------------
disves=[];
cellstartpos0=nan(3,1);
helpstartpos1=10^7; %for vessel-cell interaction; the number is here higher
than any cell volume; enables starting the while-loop
while helpstartpos1>0 %helpstartpos=0 => start position is found
try
cellstartpos0(:)= D(randi([1 length(D)]),:); % creates a random point in
the Vessel space
%to avoid cells outside the vessel space:
minx=cellstartpos0(1);
maxx=isborder(cellstartpos0(1)+lencX-1,ImX);
miny=cellstartpos0(2);
maxy=isborder(cellstartpos0(2)+lencY-1,ImY);
minz=cellstartpos0(3);
```

```
maxz=isborder(cellstartpos0(3)+lencZ-1,ImZ);
if maxx==0 || maxy==0 || maxz==0 %=>cell is partially outside the stack
helpstartpos1=10^7; %for vessel-cell interaction; the number is here higher
than any cell volume
helpstartpos2=10^7; %for cell-cell interaction
else
help1=DTStack1(miny:maxy,minx:maxx,minz:maxz); %part of the VesselMK space
colocalizing with the cell at t=0
[x,y,z]=ind2sub(size(help1), find(help1==0)); %finds zeros
help2=[x,y,z]; %coordinates of values larger 0 in help1
clear x y z
helpstartpos1= sum(ismember(B(:,:,:),help2,'rows')); %checks if vessel and
cell colocalize
end
catch
disp('error')
end
end
%build disves
cellStack1=cellStack;
cellStack1(cellStack1==0)=NaN;
helpDTStack1=DTStack1(miny:maxy,minx:maxx,minz:maxz).*cellStack1 ;
disves(1)=min(min(min(helpDTStack1)));
```

**Chemotaxis2.m**

```
%%
%============================
% cell migration
%============================
cellpos=zeros(3,length(t));
cellpos(:,1)=cellstartpos0;
cellcat(1)=0; % >0 if releasing pp, 0 if moving
for i=2:max(t)
if cellcat(i-1)==0
cellcat(i)=0;
B=B11;
cellStack=cellStack11;
stepx=normrnd(meanvelMKxy,stdvelMK)*(-1+2*(rand>.5));
stepy=normrnd(meanvelMKxy,stdvelMK)*(-1+2*(rand>.5));
stepz=normrnd(meanvelMKz,stdvelMK)*(-1+2*(rand>.5));
gradx=(DTStack1(cellpos(1,i-1)+lencX)-DTStack1(cellpos(1,i-1)))*chemf;
grady=(DTStack1(cellpos(2,i-1)+lencY)-DTStack1(cellpos(2,i-1)))*chemf;
gradz=(DTStack1(cellpos(3,i-1)+lencZ)-DTStack1(cellpos(3,i-1)))*chemf;
cellpos(1,i)=ceil(cellpos(1,i-1)+stepx+gradx);
cellpos(2,i)=ceil(cellpos(2,i-1)+stepy+grady);
cellpos(3,i)=ceil(cellpos(3,i-1)+stepz+gradz);
minx=cellpos(1,i);
maxx=isborder(cellpos(1,i)+lencX-1,ImX);
miny=cellpos(2,i);
maxy=isborder(cellpos(2,i)+lencY-1,ImY);
minz=cellpos(3,i);
maxz=isborder(cellpos(3,i)+lencZ-1,ImZ);
if maxx==0 | maxy==0 | maxz==0 | minx<1 | miny<1 | minz<1
cellpos(:,i)=cellpos(:,i-1);
disves(i)=disves(i-1);
else
cellStack1=cellStack;
cellStack1(cellStack1==0)=NaN;
```

```
help1=DTStack(miny:maxy,minx:maxx,minz:maxz).*cellStack1; %for MK_VESSEL
help3=DTStack1(miny:maxy,minx:maxx,minz:maxz).*cellStack1;
disves(i)=min(min(min(help3)));
disvesMKves(i)=min(min(min(help1)));
if disvesMKves(i)==0 && disves(i)>0 %if cell meets MK
cellpos(:,i)=cellpos(:,i-1) ; %move the cell to the last position
disves(i)=disves(i-1);
elseif disvesMKves(i)==0 && disves(i)==0 %if the cell meets a vessel
cellcat(i)=probstop(p_walk); %probability to keep walking
end
end
elseif cellcat(i-1)==1
cellpos(:,i)=cellpos(:,i-1);
disves(i)=disves(i-1);
cellcat(i)=1;
end
end
save([folderpath '\simulation1.mat'], 'aXY', 'aZ', 'cellcat', 'cellpos',
'chemf', 'chemotaxis', 'disves', 'm', 'meanvelMKxy', 'meanvelMKz',
'meanvel', 'p_walk', 'stdvel', 'stdvelMK', 'stdvelMKz', 'stepnumber')
```

## SUPPLEMENTARY FIGURES

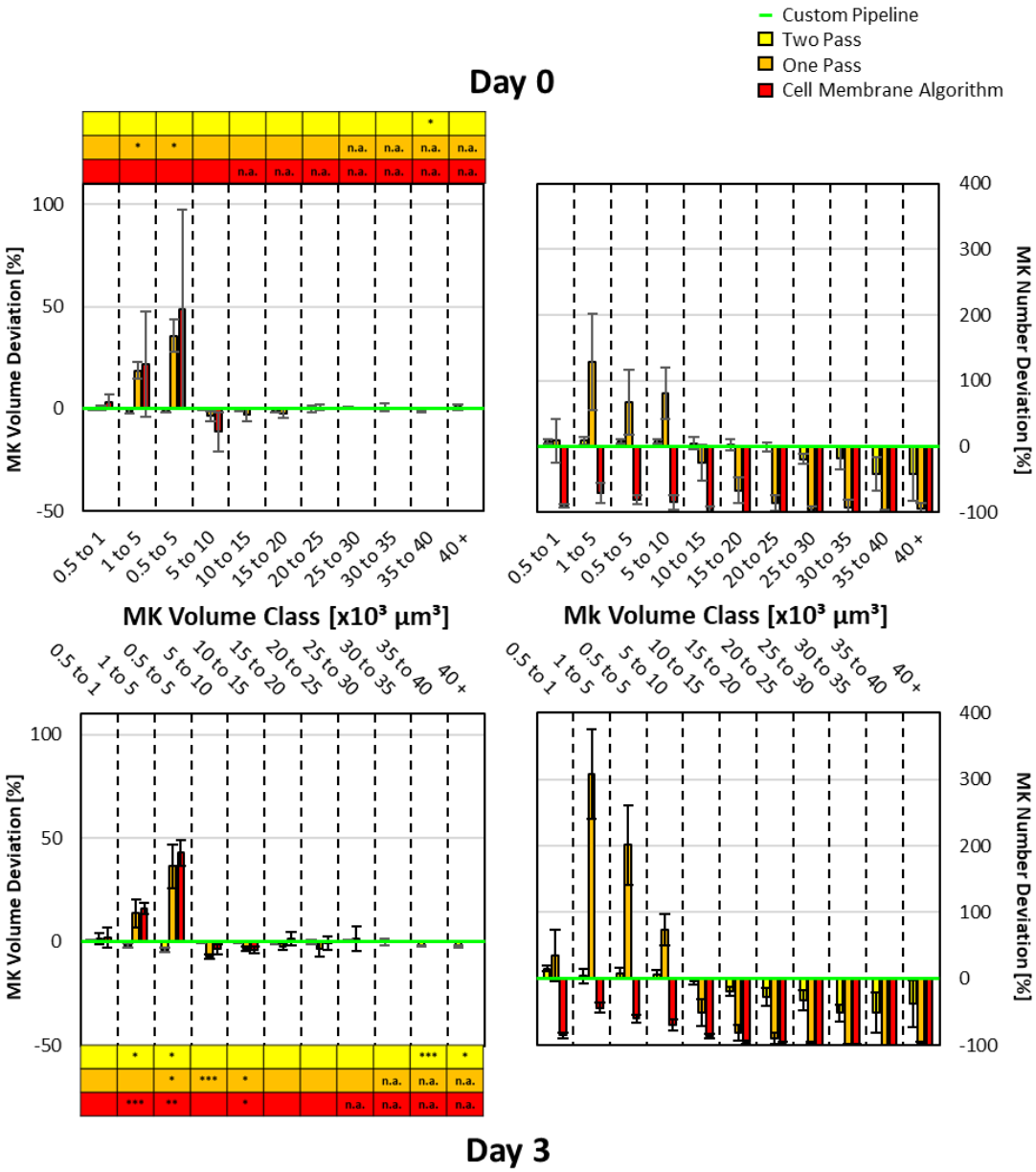


**Fig. S1. MK Volume class dependent performance of Custom Pipeline vs. Two/One pass and Cell Membrane Algorithm.** Relative MK Volume Deviation (%, left side) and Cell Number Deviation (%, right side) of two pass pipeline (yellow), one pass pipeline (orange) and Imaris Cell membrane algorithm (red) compared to custom pipeline (zero level, green). MK population binned by volume (see x-axes). Data depicted for MKs at before (day 0, top) and after (day 3, bottom) induced thrombopoiesis. The two pass pipeline generates similar results to our custom pipeline in lower and mid-tier classes, but is outperformed in larger MK classes. MK over-fragmentation is indicated using the one pass pipeline. MK numbers of larger classes are significantly underestimated, while overestimating lower classes. At the same time, mean volumes of smaller classes are significantly overestimated. Imaris cell membrane algorithm fully underperforms while underestimating numbers for all classes and highly overestimating mean volumes of lower MK classes. Bar graphs represent mean ± SD. Two-parameter T-Test: *, $p<0.05$;** $p<0.01$;*** $p<0.001$; Mann-Whitney Test: *m, $p<0.05$; Abbreviations: n.a. = insufficient sample size; blank = not significant

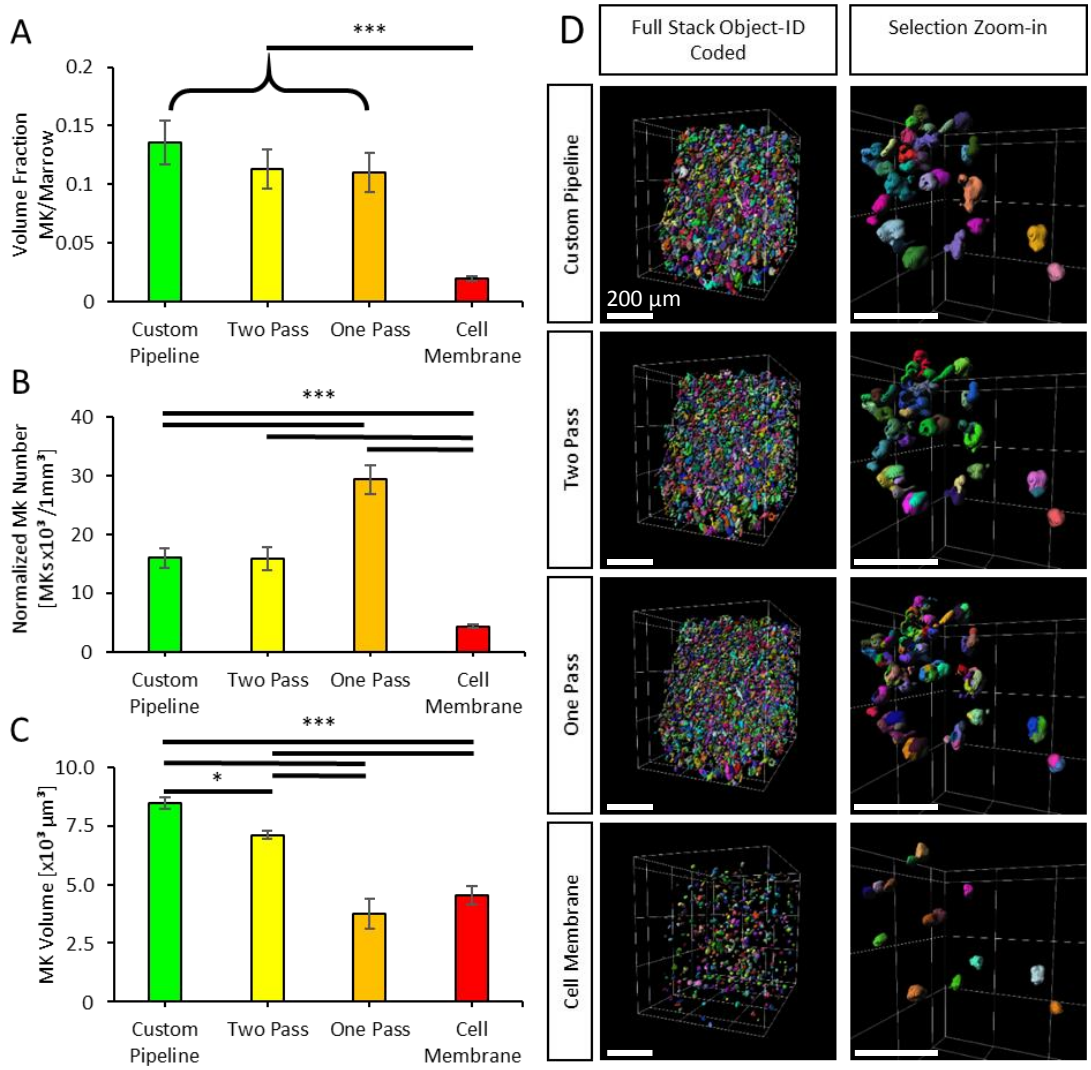


**Fig. S2. Segmentation performance of Custom Pipeline compared with One/Two Pass pipeline and Imaris Cell Membrane Algorithm at day 3 after platelet depletion.** (**A-C**) Data depicted for custom pipeline (green bar), two pass pipeline (yellow bar), one pass pipeline (orange bar), and Imaris cell membrane algorithm (red bar) after induced thrombopoiesis (day 3). (**A**) MK to marrow volume fraction is comparable between custom and one /two pass pipelines with a massive drop when using Imaris cell membrane algorithm. (**B**) Normalized mean MK numbers are comparable between the custom and the two pass pipeline; contrary to significant increase with the one pass pipeline and massive drop with Imaris cell membrane algorithm. (**C**) Mean MK volumes significant decrease compared to custom pipeline. Bar graphs represent mean ± SD. *, $p<0.05$;** $p<0.01$;*** $p<0.001$ (One way ANOVA, Tukey post-hoc test). (**D**) Exemplified segmentation results. Left column: Full stack with segmented MKs, object ID color-coded. Sparsely located MKs with cell membrane algorithm opposed to other pipelines. Right column: Zoom-in to selection. MK fragmentation increases in one and two pass compared to custom pipeline. Cell membrane algorithm with sparse and small MKs. Grid and scale bar size = 200 µm.

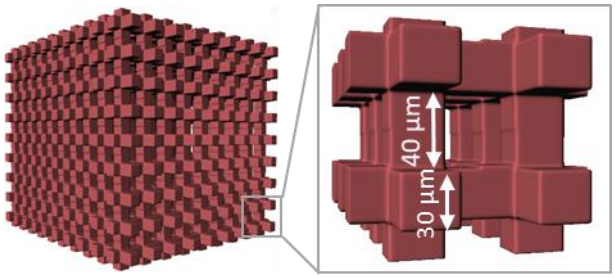


**Fig. S3. Generation of artificial vascular network for computational simulations.** An artificial vessel network was designed with a vessel size of 30 µm and intravascular distance of 40 µm mimicking the physiological structures.

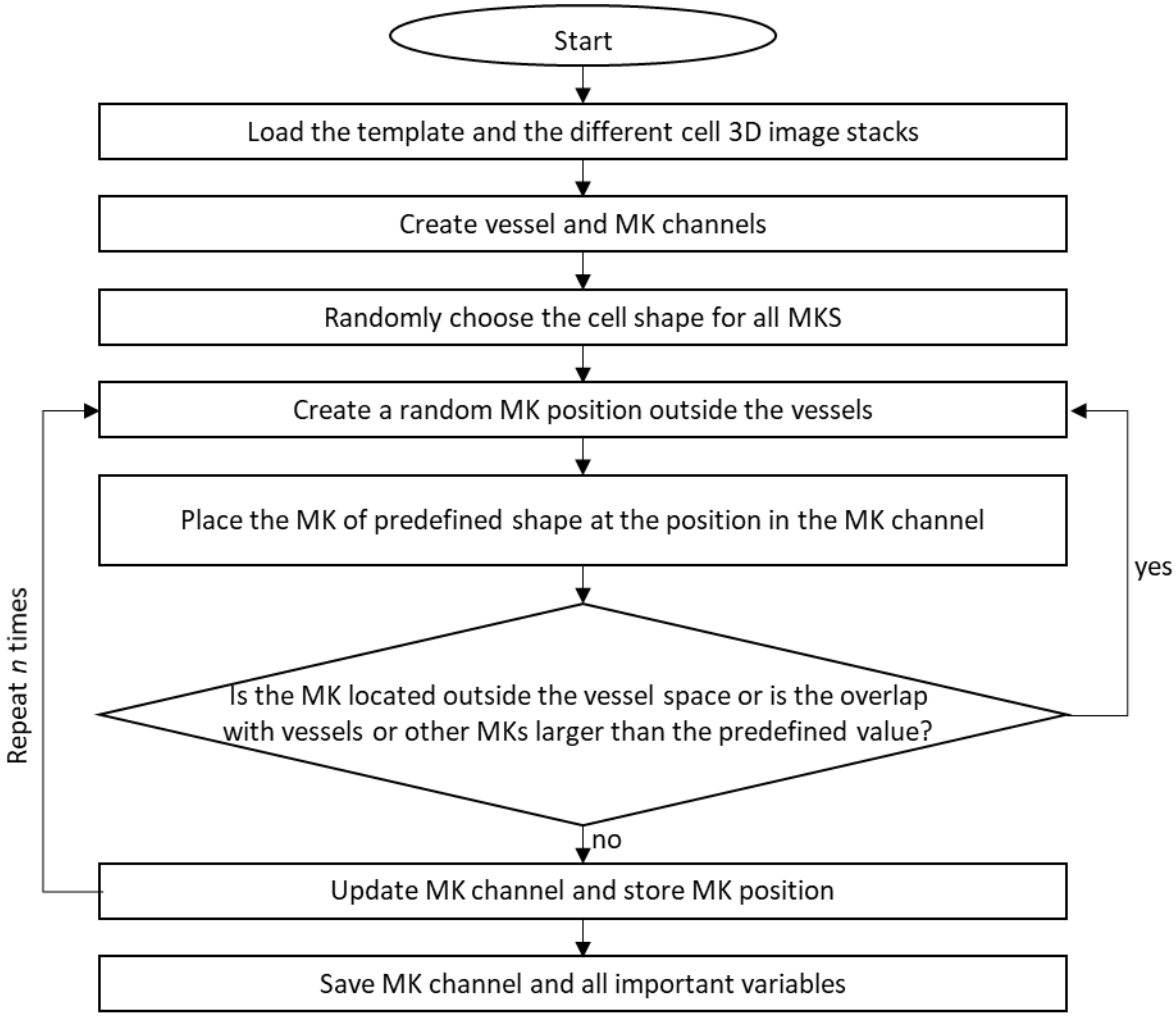


**Fig. S4. Algorithm for simulation of megakaryocyte distribution in the bone marrow.** 3D microscopy image stacks of the vasculature and single MKs are loaded and converted into 3D tensors and a virtual space for MKs is created. MKs are placed cell by cell into the MK space in a way that the colocalization with the vasculature or other MKs does not exceed a predefined value.

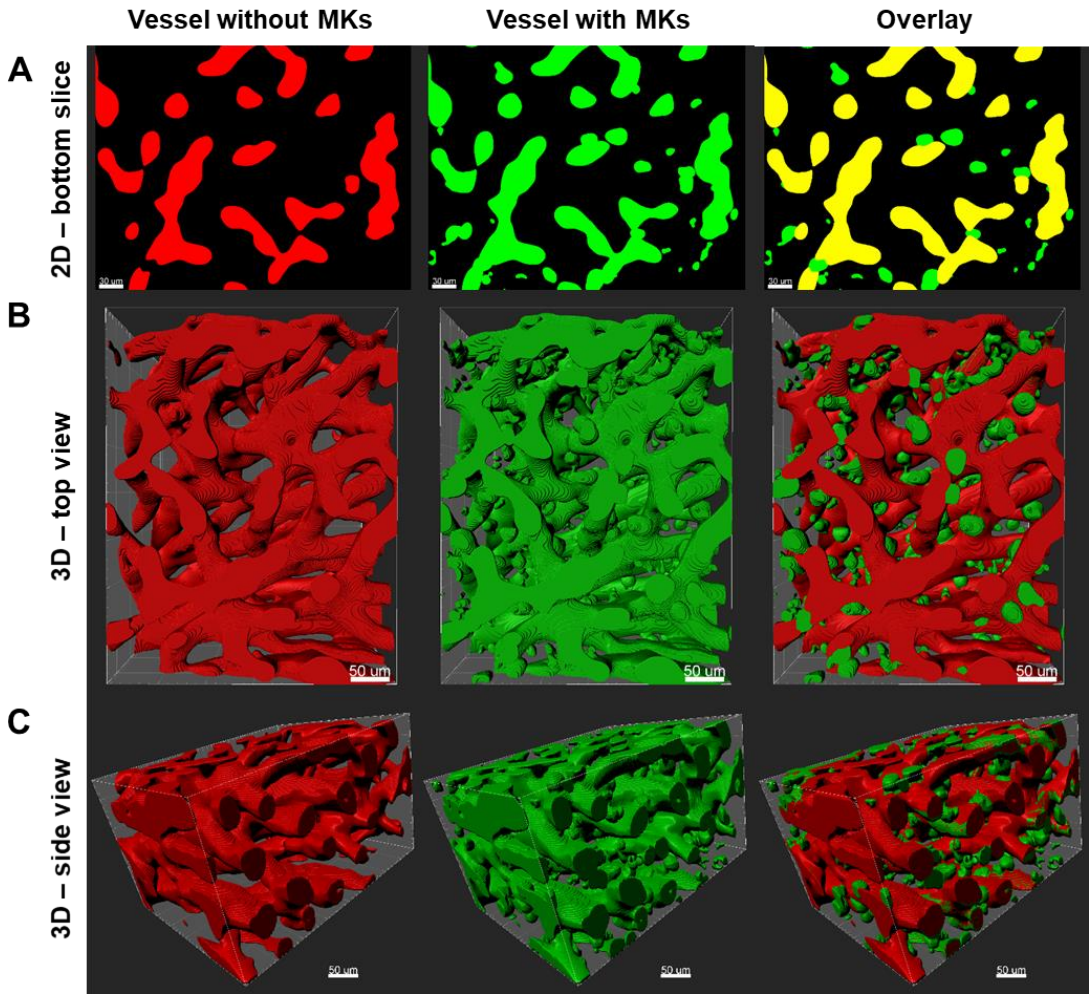


**Figure S5. Segmented 3D-image of the BM featuring the vasculature and the MKs.** This 3D image served as a template for static simulations of MKs and dynamic simulations of Neutrophil and HSC migration. Left: vasculature without MKs, middle: the vasculature with MKs, right: overlay. Scale bar for (A) is 30 µm, scale bar and grid size for (B-C) 50 µm.

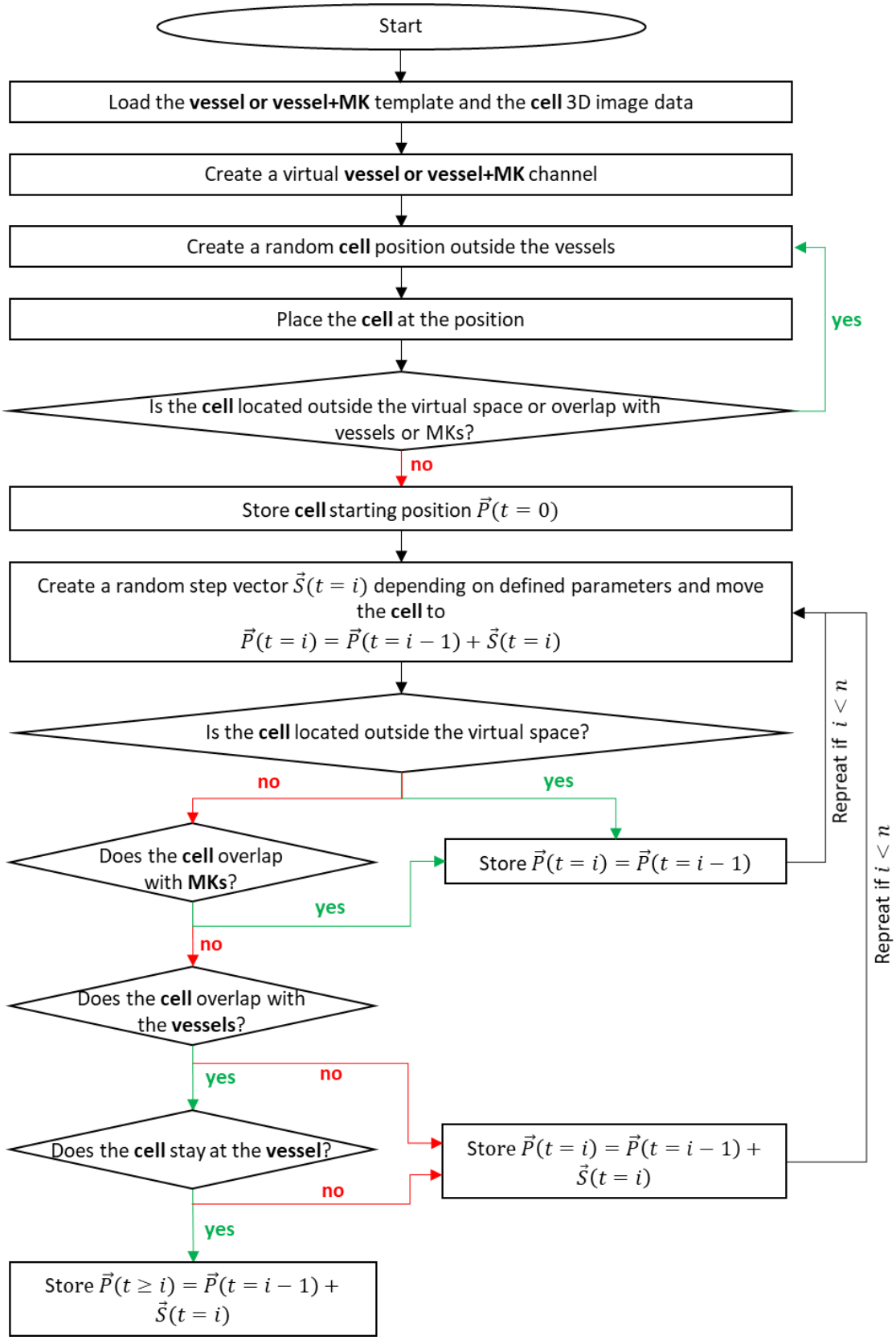


**Fig. S6. Algorithm for simulation of cell migration in the bone marrow.** 3D microscopy image stacks of the vasculature (and MKs) and images of migrating cells are loaded and converted into 3D. The migrating cell is placed into the template in a way that the colocalization with the vasculature or other MKs does not exceed a predefined value. Cell migration is performed with steps of random size in all directions or according to virtual chemotaxis and cell-vessel adhesion effects.

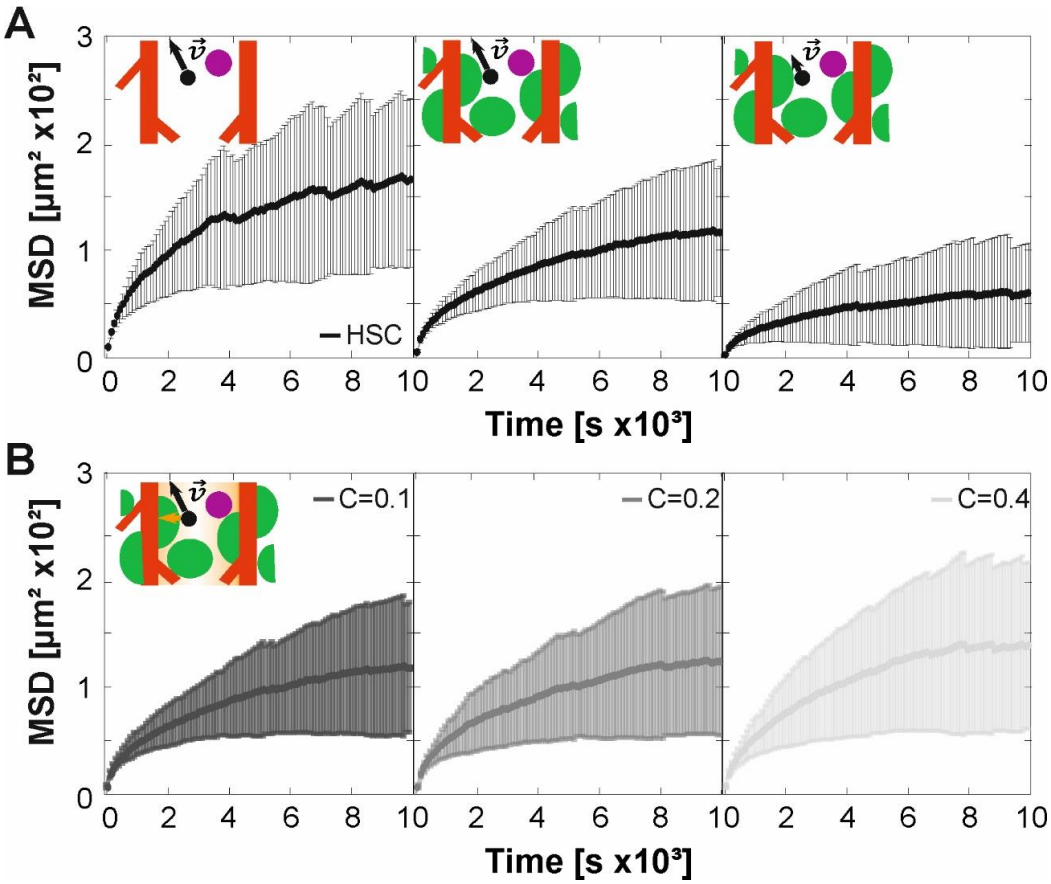


**Fig. S7. Adjustable biophysical parameters influence simulated cell migration in the BM.** Individual plots for the Simulation of cell migration in bone marrow with adjustable parameters: cell type, bone marrow crowdedness, cell velocity, chemotaxis and vessel stickiness was performed. Mean squared displacement (MSD) of HSC trajectories for the different simulation conditions. (**A**) MK-free template at high velocities (left), MK-containing template at high (middle) and low velocities (right). $P_{EV}$ = 100% and C = 0. (**B**) MSD analysis of HSC migration data for $P_{EV}$ = 100% for increasing chemotaxis parameter C = [0; 0.1; 0.2; 0.4] in presence of MKs.

**Table S1: Comparison of image stacks versus virtual slices as well as different segmentation pipelines.** Data depicted as mean ± SD.

| **Comparison of full image stacks and virtual slices** | | | | |
|---|---|---|---|---|
| | **Day 0** | | **Day 3** | |
| | **Stack** | **Slice** | **Stack** | **Slice** |
| MK-vessel distance [µm] | 2.4 ± 0.5 | 6.7 ± 1.1 | 1.1 ± 0.2 | 4.1 ± 0.7 |
| Vessel interspace [µm] | 43.1 ± 1.9 | 50.4 ± 2.6 | 40.8 ± 1.4 | 45.8 ± 1.0 |
| MK diameter [µm] | 26.1 ± 1.9 | 11.9 ± 0.1 | 32.8 ± 0.9 | 14.4 ± 1.1 |

| **Comparison of segmentation pipelines** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Day 0** | | | | **Day 3** | | | |
| | **Custom Pipeline** | **Two Pass** | **One Pass** | **Cell Memb.** | **Custom Pipeline** | **Two Pass** | **One Pass** | **Cell Memb.** |
| MK/marrow vol. fraction | 0.11 ± 0.02 | 0.11 ± 0.02 | 0.09 ± 0.01 | 0.01 ± 0.01 | 0.14 ± 0.02 | 0.11 ± 0.02 | 0.11 ± 0.02 | 0.01 ± 0.00 |
| MK number [MKs/1mm³ marrow volume] | 20800.6 ± 4224.7 | 22003.1 ± 4386.3 | 29401.3 ± 939.3 | 3227.8 ± 1589.1 | 15991.8 ± 1699.7 | 15868.6 ± 2047.7 | 29353.5 ± 2452.0 | 4343.7 ± 246.4 |
| MK volume [µm³] | 5206.0 ± 71.8 | 4843.8 ± 77.1 | 3392.2 ± 618.0 | 2664.8 ± 383.12 | 8458.4 ± 256.5 | 7114.8 ± 152.76 | 3765.0 ± 646.6 | 4539.6 ± 383.1 |

**Table S2. Number of steps until vessel entrance.** HSC: hematopoietic stem cell; Prob-EV: probability for the cell to enter the vessel after a contact with the vasculature; $V^{high}$: high velocity of v = 3 ± 2 µm/step; $V^{low}$: low velocity of v = 2 ± 1 µm/step; C: parameter for chemotaxis, where for C = 0 the effect is absent. Data are presented as mean ± SD.

| | | | Without MKs | | With MKs | |
|---|---|---|---|---|---|---|
| | | | HSC | Neutrophil | HSC | Neutrophil |
| Prob-EV = 100% | $V^{high}$ | C = 0 | 513 ± 1107 | 628 ± 1270 | 8057 ± 10310 | 7869 ± 10175 |
| | | C = 0.1 | 346 ± 850 | 228 ± 442 | 4988 ± 7763 | 2553 ± 4384 |
| | $V^{low}$ | C = 0 | 8100 ± 13162 | 7406 ± 12767 | 19694 ± 13924 | 20541 ± 13626 |
| | | C = 0.1 | 6807 ± 12390 | 4915 ± 10958 | 18404 ± 14299 | 16160 ± 14499 |
| Prob-EV = 50% | $V^{high}$ | C = 0 | 625 ± 1315 | 654 ± 1172 | 10753 ± 11606 | 8783 ± 10645 |
| | | C = 0.1 | 317 ± 560 | 309 ± 545 | 7291 ± 9450 | 3944 ± 5645 |
| | $V^{low}$ | C = 0 | 7862 ± 13028 | 8015 ± 13100 | 21675 ± 13164 | 21872 ± 13068 |
| | | C = 0.1 | 6900 ± 12472 | 5225 ± 11146 | 20121 ± 13832 | 16107 ± 14415 |

**Table S3. Saturation limit fit of MSD(t) curves.** Simulated and experimental MSD(t) curves were fitted using equation 2. Please note that the parameter *R* (response range) and *r* (response rate) do not have a physical meaning. For *in vivo* measurement, the fit results of the average MSD(t) (Figure 5C-D) is shown.

| | *Limit* | ± | *R* | ± | *r* | ± | $\chi_r^2$ | |
|---|---|---|---|---|---|---|---|---|
| ***in vivo measurements*** | | | | | | | | |
| control | 120 | 5.58 | 115 | 5.41 | 0.964 | 0.003 | 0.997 | |
| depleted | 66.7 | 1.22 | 63.1 | 1.10 | 0.934 | 0.003 | 0.997 | |
| ***Simulation results*** | | | | | | | | |
| HSC (MK-free, $v^{high}$) | 171 | 1.3 | 135 | 2.1 | 0.735 | 0.009 | 0.982 | Prob-EV= 100%, C= 0 |
| Neutrophil (MK-free, $v^{high}$) | 182 | 1.8 | 145 | 1.8 | 0.784 | 0.008 | 0.985 | |
| HSC (+MKs, $v^{high}$) | 133 | 0.8 | 106 | 0.7 | 0.804 | 0.004 | 0.996 | |
| Neutrophil (+MKs, $v^{high}$) | 116 | 0.5 | 89.2 | 0.7 | 0.749 | 0.005 | 0.994 | |
| HSC (+MKs, $v^{low}$) | 66.7 | 1.0 | 50.6 | 0.9 | 0.791 | 0.011 | 0.971 | |
| Neutrophil (+MKs, $v^{low}$) | 68.0 | 0.4 | 57.4 | 0.7 | 0.740 | 0.007 | 0.989 | |
| HSC, C=0.1 | 151 | 1.0 | 133 | 1.5 | 0.783 | 0.006 | 0.990 | Prob-EV = 100% |
| HSC, C=0.2 | 140 | 1.2 | 111 | 0.9 | 0.805 | 0.005 | 0.993 | |
| HSC, C=0.4 | 135 | 0.9 | 108 | 0.7 | 0.809 | 0.004 | 0.996 | |
| HSC (+MKs, $v^{high}$) | 138 | 2.1 | 107 | 1.5 | 0.852 | 0.006 | 0.992 | Prob-EV = 50%, C=0 |
| Neutrophil (+MKs, $v^{high}$) | 115 | 0.5 | 92.1 | 0.8 | 0.741 | 0.005 | 0.994 | |

**Table S4. 3D anomalous diffusion fit of MSD(t) curves.** The first 25 % of simulated and experimental MSD(t) curves were fitted using equation 3. For *in vivo* measurement, the fit results of the average MSD(t) (Figure 5C-D) is shown.

| | $D_{app}$ | ± | $\alpha$ | ± | $\chi_r^2$ | |
|---|---|---|---|---|---|---|
| ***in vivo measurements*** | | | | | | |
| control | 1.070 | 0.038 | 0.678 | 0.018 | 0.995 | |
| depleted | 0.854 | 0.043 | 0.716 | 0.026 | 0.973 | |
| ***Simulation results*** | | | | | | |
| HSC (MK-free, $v^{high}$) | 8.9 | 0.04 | 0.479 | 0.007 | 0.996 | Prob-EV= 100%, C= 0 |
| Neutrophil (MK-free, $v^{high}$) | 8.5 | 0.05 | 0.467 | 0.009 | 0.995 | |
| HSC (+MKs, $v^{high}$) | 6.1 | 0.03 | 0.426 | 0.007 | 0.996 | |
| Neutrophil (+MKs, $v^{high}$) | 6.2 | 0.02 | 0.440 | 0.005 | 0.998 | |
| HSC (+MKs, $v^{low}$) | 3.3 | 0.02 | 0.423 | 0.009 | 0.993 | |
| Neutrophil (+MKs, $v^{low}$) | 3.4 | 0.02 | 0.427 | 0.010 | 0.990 | |
| HSC, C=0.1 | 6.7 | 0.04 | 0.520 | 0.007 | 0.997 | Prob-EV = 100% |
| HSC, C=0.2 | 6.4 | 0.05 | 0.446 | 0.010 | 0.991 | |
| HSC, C=0.4 | 6.1 | 0.04 | 0.403 | 0.010 | 0.991 | |
| HSC (+MKs, $v^{high}$) | 5.8 | 0.02 | 0.414 | 0.007 | 0.996 | Prob-EV = 50%, C=0 |
| Neutrophil (+MKs, $v^{high}$) | 6.0 | 0.03 | 0.435 | 0.006 | 0.997 | |