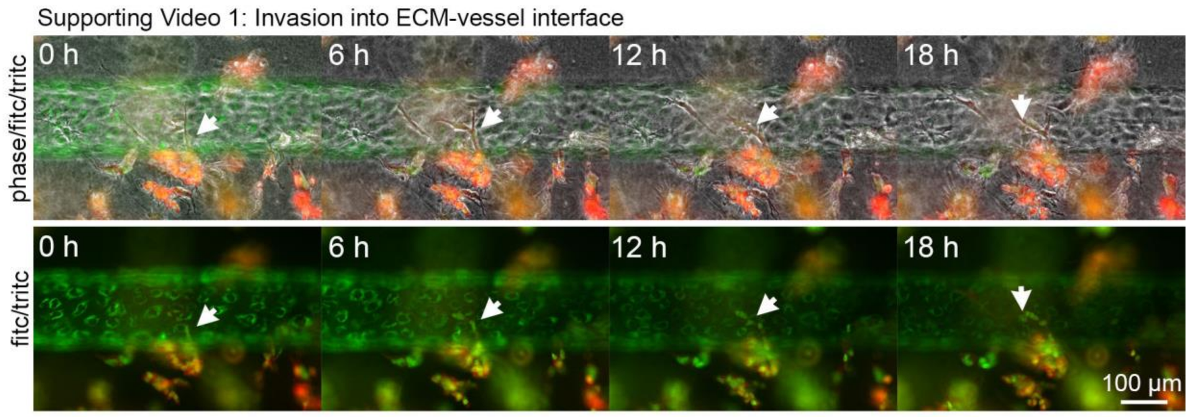
Mitosis-mediated intravasation in a tissue-engineered tumor-microvessel platform

Andrew D. Wong[1,2] and Peter C. Searson[1,2]
[1] Department of Materials Science and Engineering, Johns Hopkins University, 3400 N. Charles St., Baltimore, Maryland 21218, USA.
[2] Institute for Nanobiotechnology (INBT), 100 Croft Hall, Johns Hopkins University, 3400 N. Charles St., Baltimore, Maryland 21218, USA.

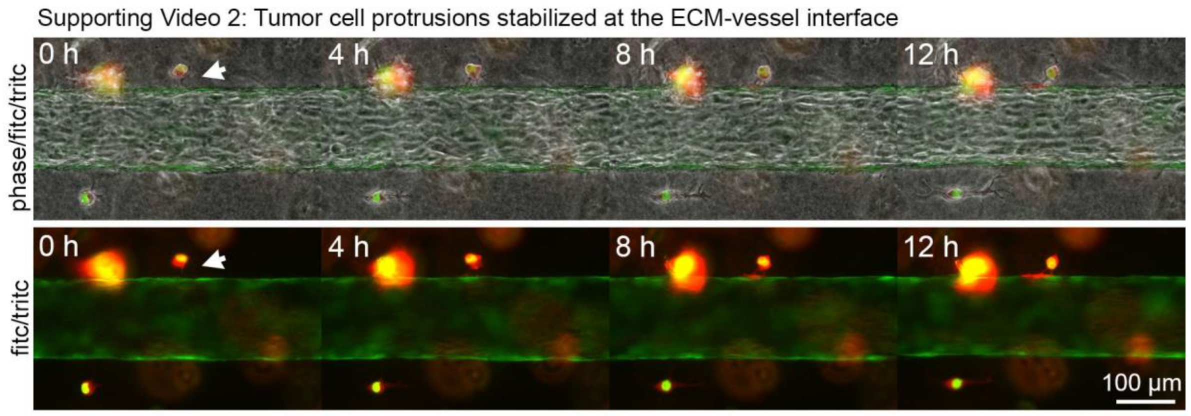
SUPPLEMENTARY VIDEO LEGENDS & STILLS



Supplementary Video S1.

Title: Collective migration and invasion of MDA-MB-231 breast cancer cells (BCCs) into the ECM-vessel interface.

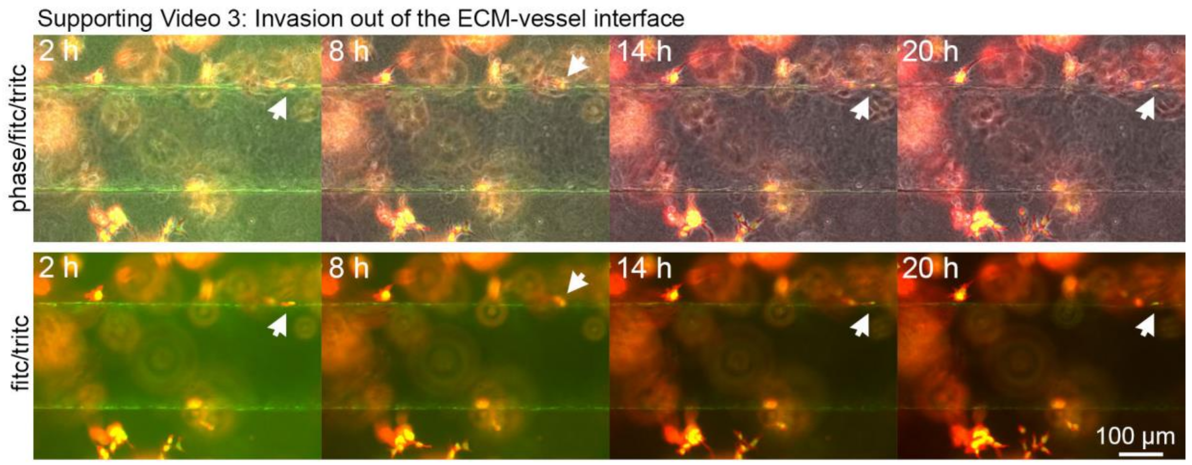
Legend: Multiple BCCs from clusters located along the vessel "stream" into the interface along the same collagen degraded track (arrows). BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HMVECs. The imaging plane is focused on the bottom pole of the vessel. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 2: Tumor cell protrusions stabilized at the ECM-vessel interface
phase/fitc/tritc
fitc/tritc
0 h
4 h
8 h
12 h
100 µm

Supplementary Video S2.

Title: Extended protrusions/invadopodia from an MDA-MB-231 breast cancer cell (BCC) are stabilized at the ECM-vessel interface.

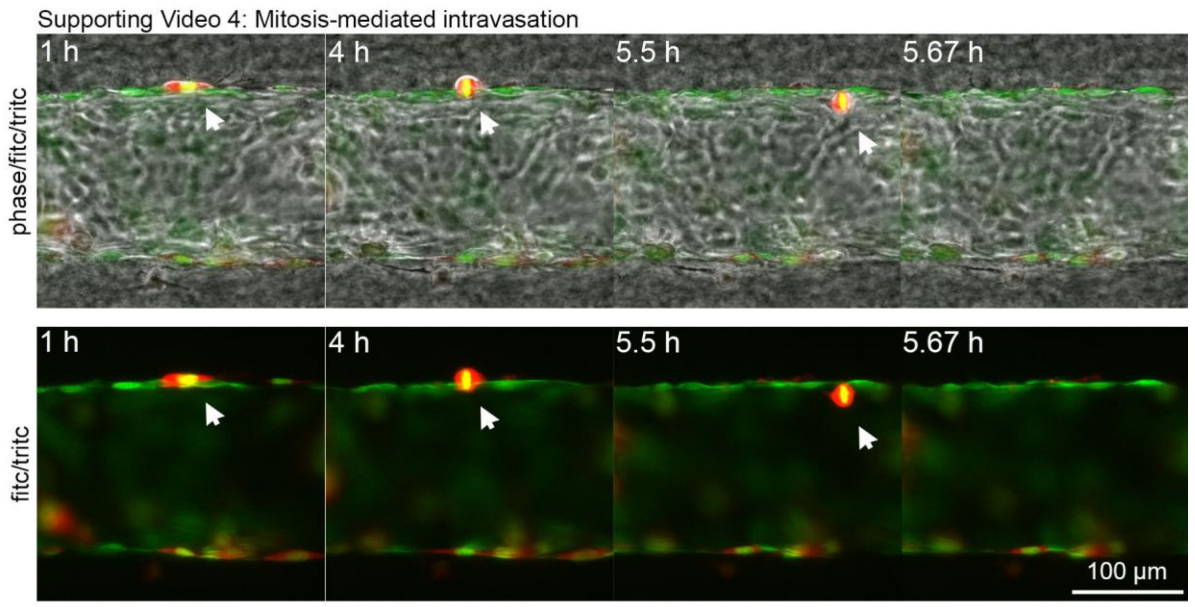
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 3: Invasion out of the ECM-vessel interface
phase/fitc/tritc
fitc/tritc
2 h
8 h
14 h
20 h
100 μm

Supplementary Video S3.

Title: Entry and exit from the ECM-vessel interface of MDA-MB-231 breast cancer cells (BCC).

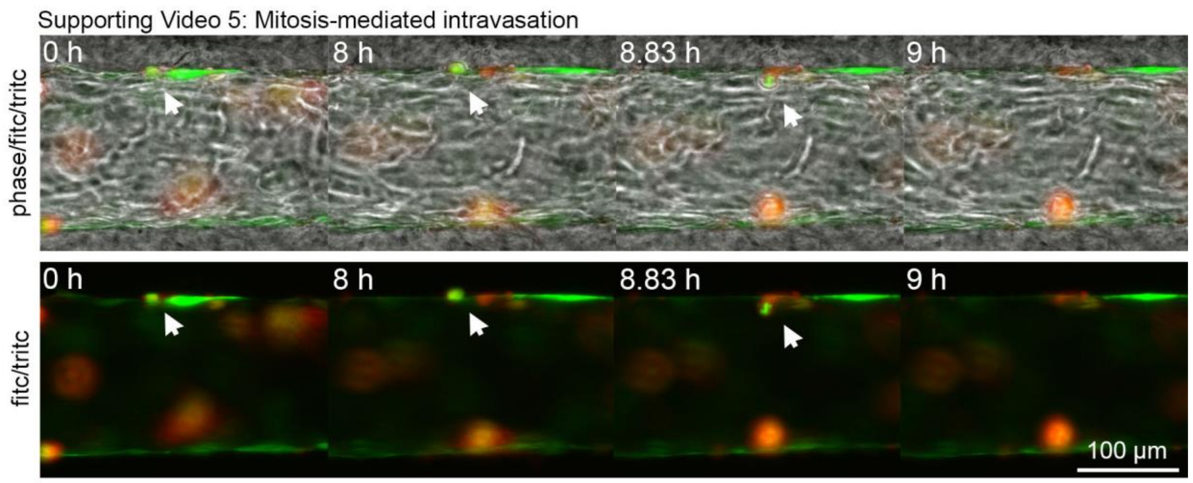
Legend: BCCs are able to repeatedly enter and exit the ECM-vessel interface due to a pre-defined etched track seen in phase-contrast. BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HUVECs. The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.



Supplementary Video S4.

Title: Mitosis-mediated intravasation of an MDA-MB-231 breast cancer cell (BCC) (#3).

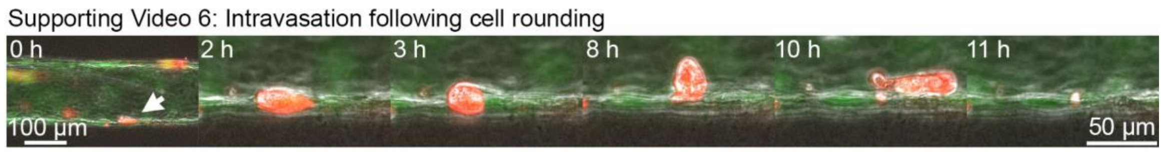
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.



Supplementary Video S5.

Title: Mitosis-mediated intravasation of an MDA-MB-231 breast cancer cell (BCC) (#5).

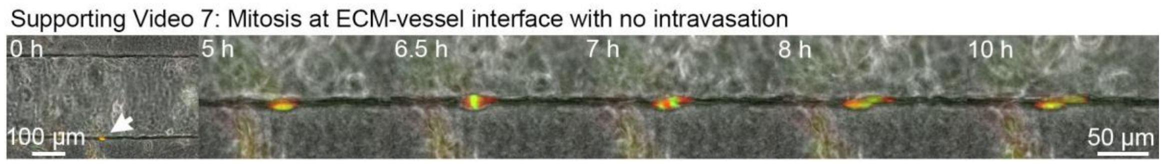
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.



Supplementary Video S6.

Title: Intravasation of an RFP expressing MDA-MB-231 breast cancer cell (BCC) where cell rounding enables transendothelial migration.

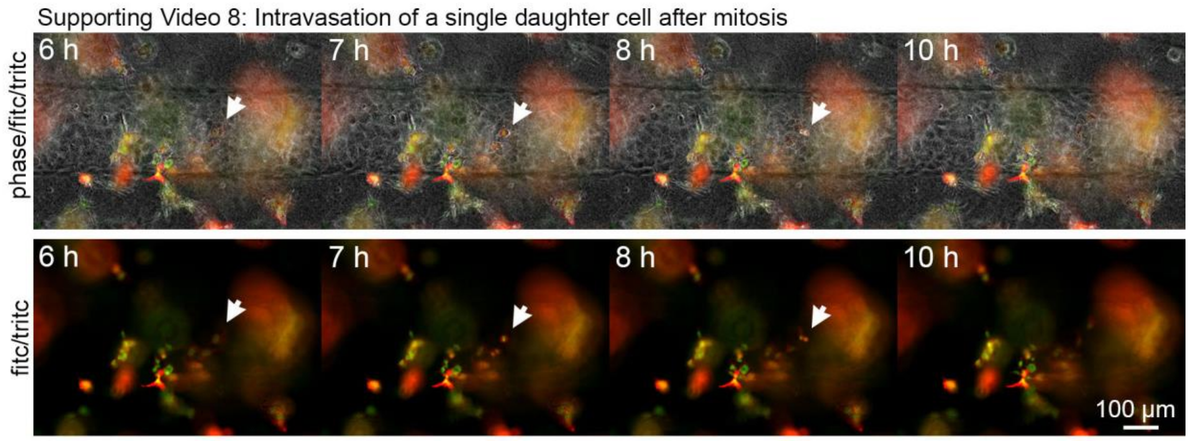
Legend: BCC expresses RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 7: Mitosis at ECM-vessel interface with no intravasation
0 h
5 h
6.5 h
7 h
8 h
10 h
100 µm
50 µm

Supplementary Video S7.

Title: Mitosis of an MDA-MB-231 breast cancer cell (BCC) at the ECM-vessel interface without intravasation.

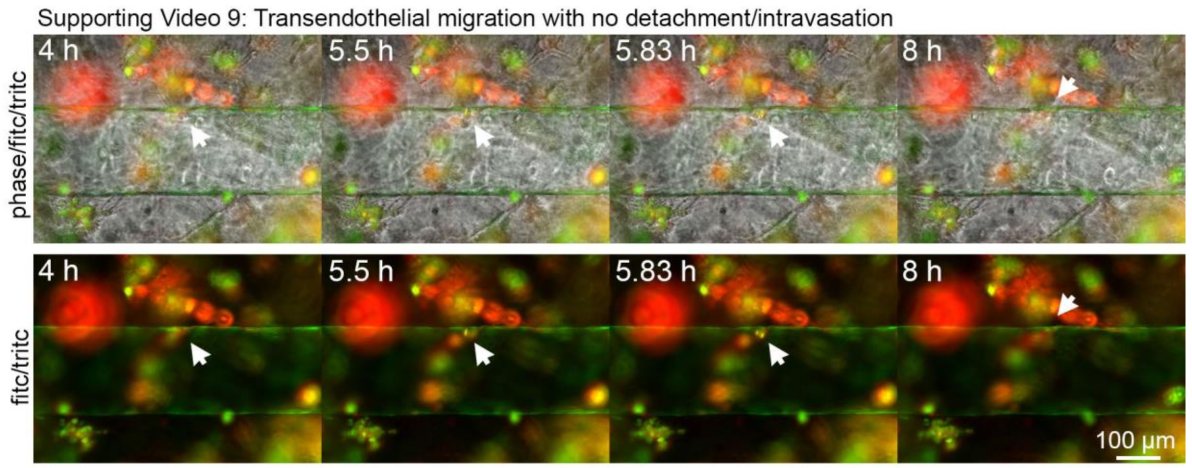
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HUVECs. The imaging plane is focused on the vessel equator. Phase-contrast with fluorescence overlays (left) are displayed. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 8: Intravasation of a single daughter cell after mitosis
phase/fitc/tritc
fitc/tritc
6 h
7 h
8 h
10 h
100 µm

Supplementary Video S8.

Title: Mitosis-mediated intravasation of an MDA-MB-231 breast cancer cell (BCC) where one daughter cell intravasates and the other returns into the ECM-vessel interface.

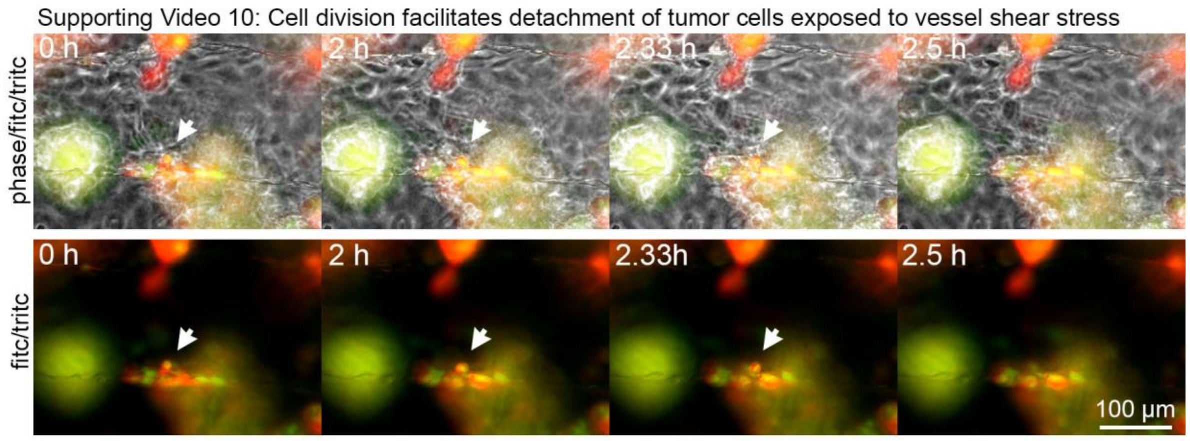
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HUVECs. The imaging plane is focused at the vessel pole. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 9: Transendothelial migration with no detachment/intravasation
phase/fitc/tritc
fitc/tritc
4 h
5.5 h
5.83 h
8 h
100 µm

Supplementary Video S9.

Title: Transendothelial migration of an MDA-MB-231 breast cancer cell (BCC) following cell division without detaching into vessel flow.

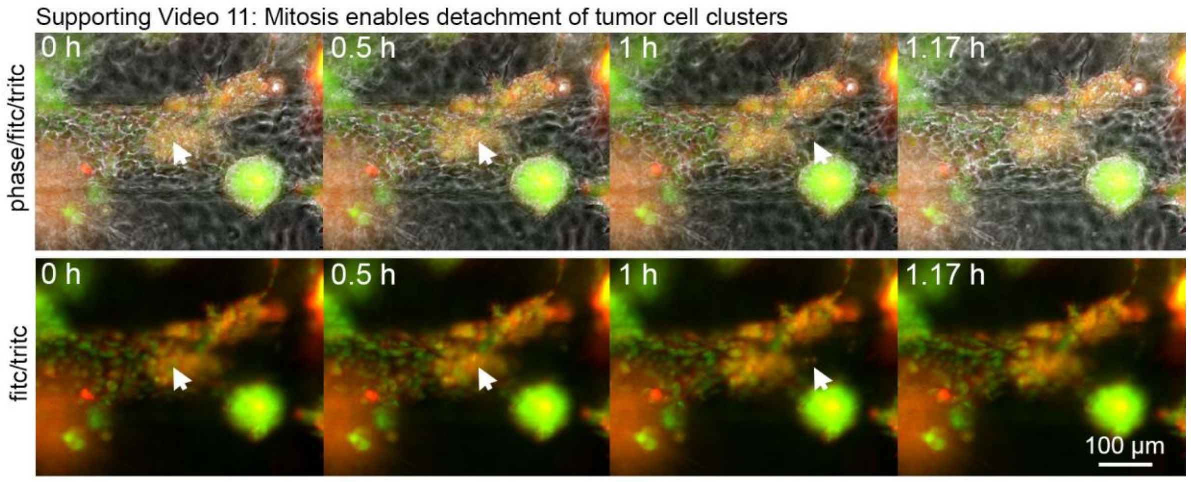
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused at the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 10: Cell division facilitates detachment of tumor cells exposed to vessel shear stress
phase/fitc/tritc
fitc/tritc
0 h
2 h
2.33 h
2.5 h
0 h
2 h
2.33h
2.5 h
100 µm

Supplementary Video S10.

Title: Single MDA-MB-231 breast cancer cell (BCC) detachment from a collectively invaded portion of the vessel.

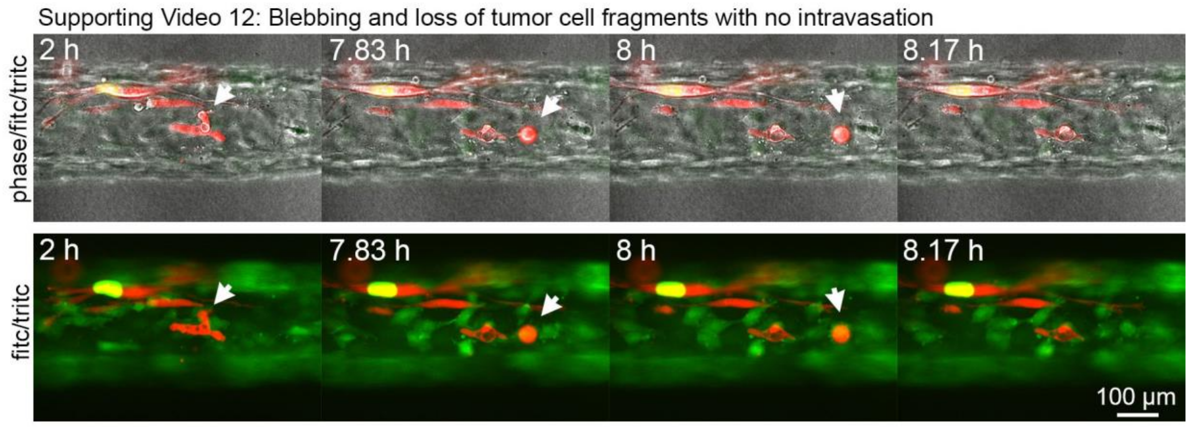
Legend: The BCC undergoes mitosis during detachment. BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HMVEC. The imaging plane is focused at the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 11: Mitosis enables detachment of tumor cell clusters
phase/fitc/tritc
fitc/tritc
0 h
0.5 h
1 h
1.17 h
0 h
0.5 h
1 h
1.17 h
100 µm

Supplementary Video S11.

Title: Multiple MDA-MB-231 breast cancer cells (BCC) detaching from a collectively invaded portion of the vessel.

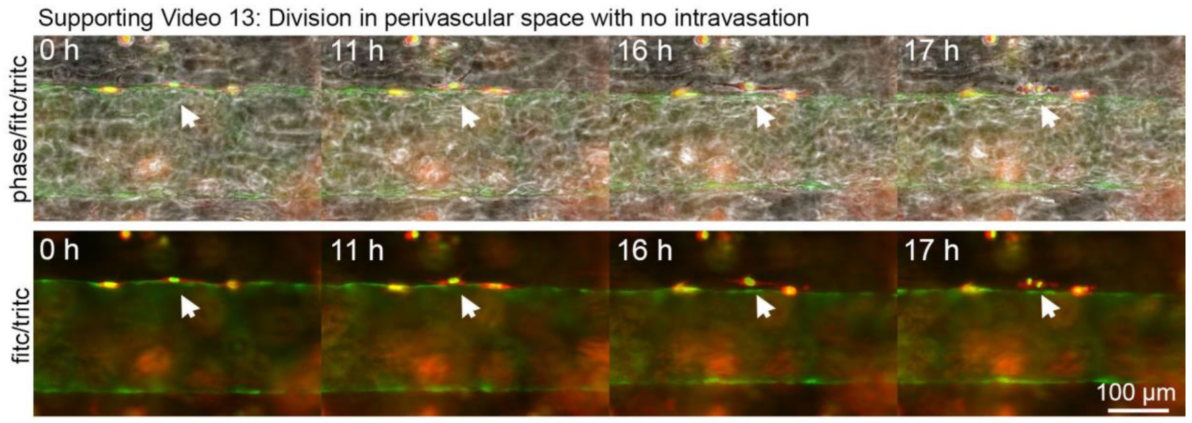
Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of HMVEC. The imaging plane is focused at the vessel pole. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.


Supporting Video 12: Blebbing and loss of tumor cell fragments with no intravasation
phase/fitc/tritc
fitc/tritc
2 h
7.83 h
8 h
8.17 h
100 µm

Supplementary Video S12.

Title: Shedding of blebs/protrusions from MDA-MB-231 breast cancer cells (BCC) at the ECM-vessel interface.

Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused at the vessel pole. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.

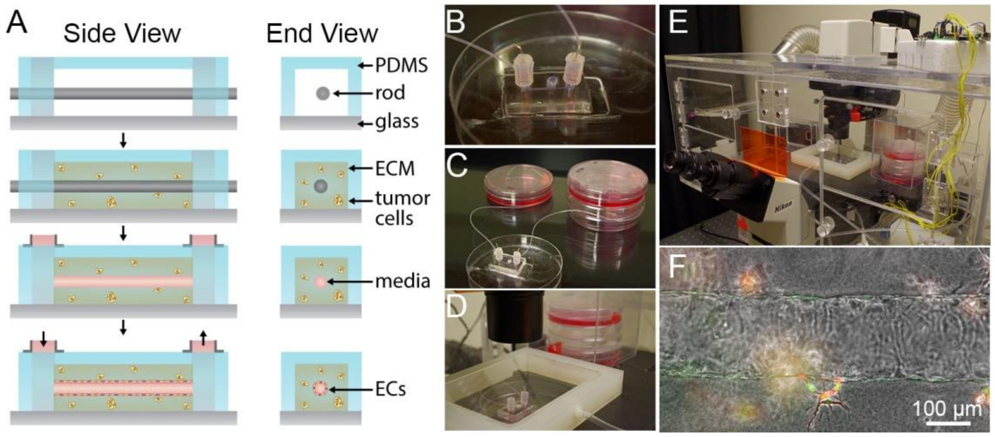

Supporting Video 13: Division in perivascular space with no intravasation
phase/fitc/tritc
fitc/tritc
0 h
11 h
16 h
17 h
100 µm

Supplementary Video S13.

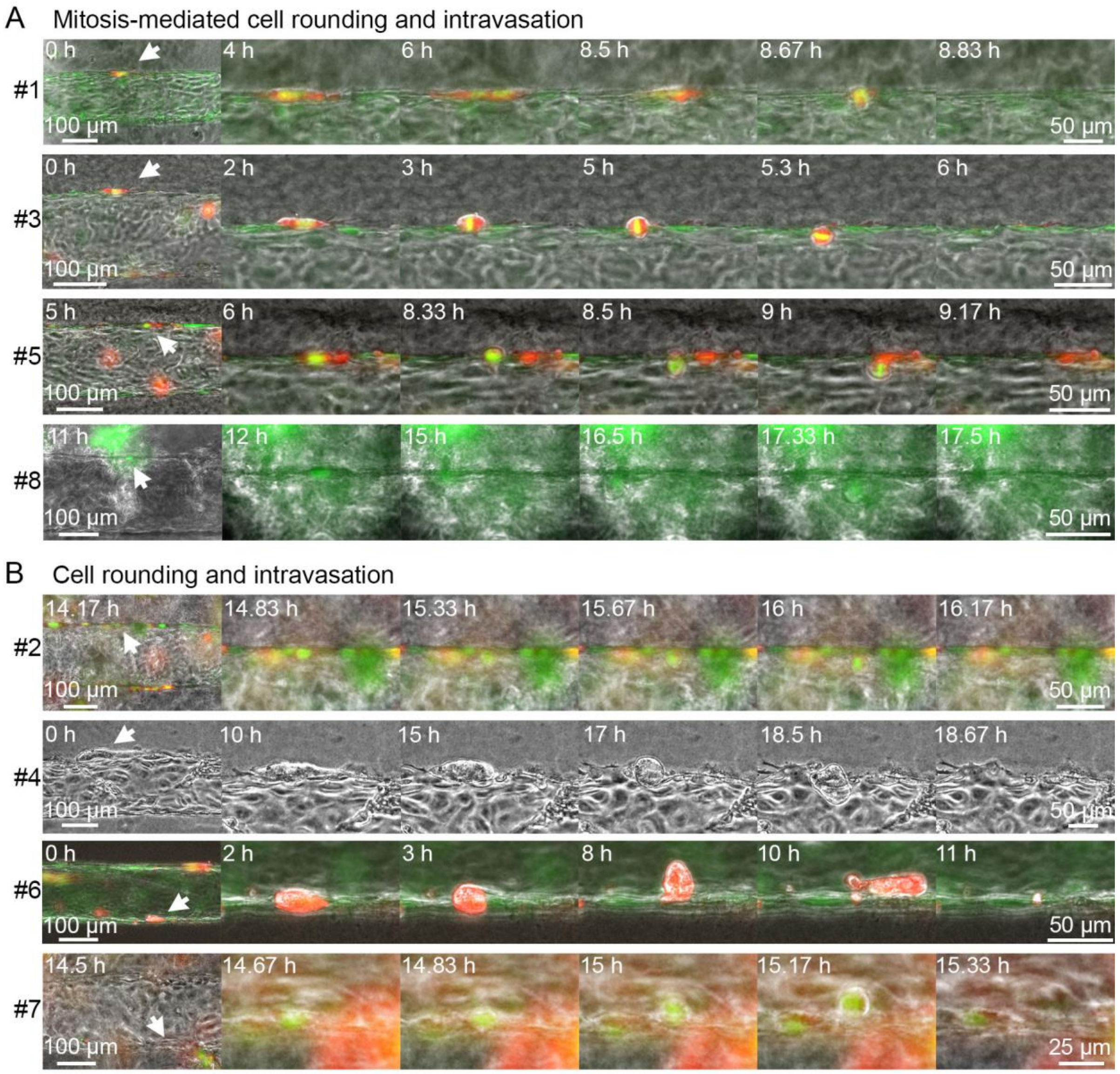
Title: Perivascular space reduces deflection from dividing MDA-MB-231 breast cancer cells (BCC) at the ECM-vessel interface.

Legend: BCCs express GFP in the nucleus (green) and RFP in the cytoplasm (red). The vessel endothelium is comprised of VeraVec-GFP cells (green). The imaging plane is focused at the vessel equator. Phase-contrast with fluorescence overlays (left) and fluorescence overlays (right) are displayed side-by-side. Images are captured every 10 minutes. Flow is from left to right.

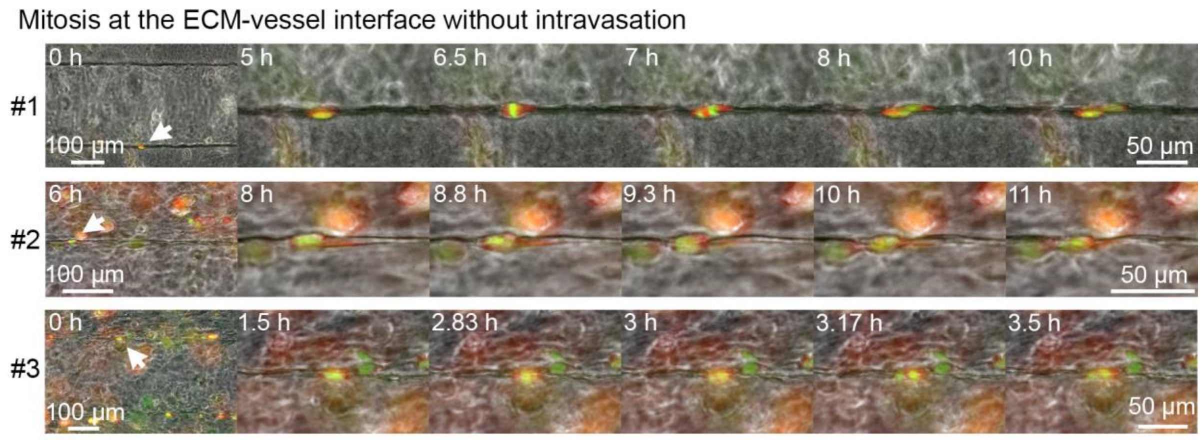
SUPPLEMENTARY FIGURES



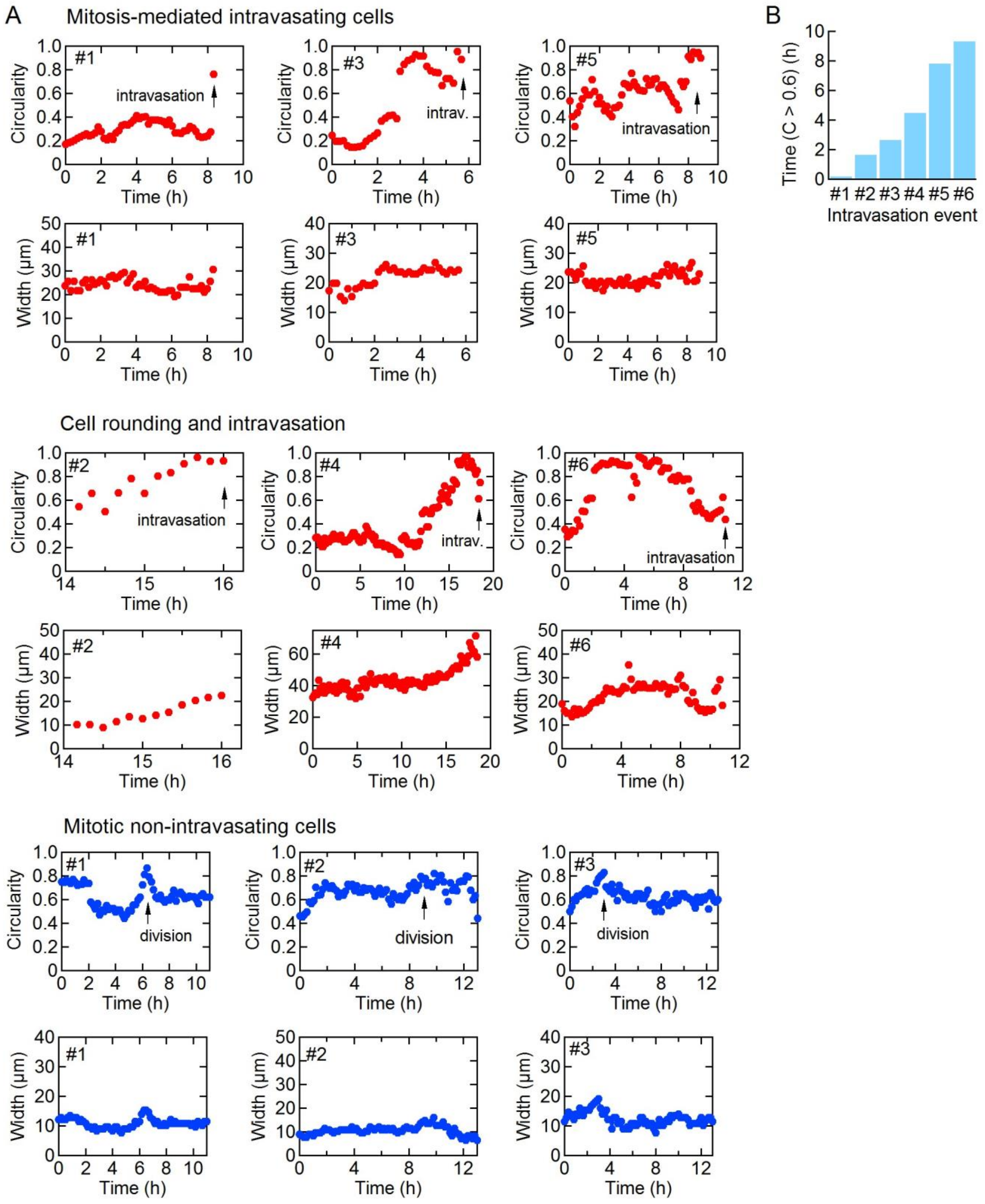
Supplementary Figure S1. Fabrication and imaging of the tumor-microvessel platform. **A,** schematic illustration of microvessel fabrication. Briefly, the housing for the microvessel is made of polydimethylsiloxane (PDMS) cast in an aluminum mold and modified for connections with external tubing. A cylindrical template rod with a diameter of 150 μm is positioned inside the housing. A solution form of the extracellular matrix (ECM), collagen type I, is introduced around the template rod. After gelation of the ECM, the rod is removed, leaving behind an empty cylindrical channel that is subsequently lined with endothelial cells. Single or clusters of tumor cells are mixed into the ECM solution prior to its injection around the template rod. **B,** device connected to flow. **C,** platform on bench top. **D,** platform mounted in live-cell chambers. **E,** live-cell imaging setup on microscope. **F,** Phase-contrast image with fluorescence overlays of HUVEC microvessel and invading dual labeled breast cancer cells (BCCs).



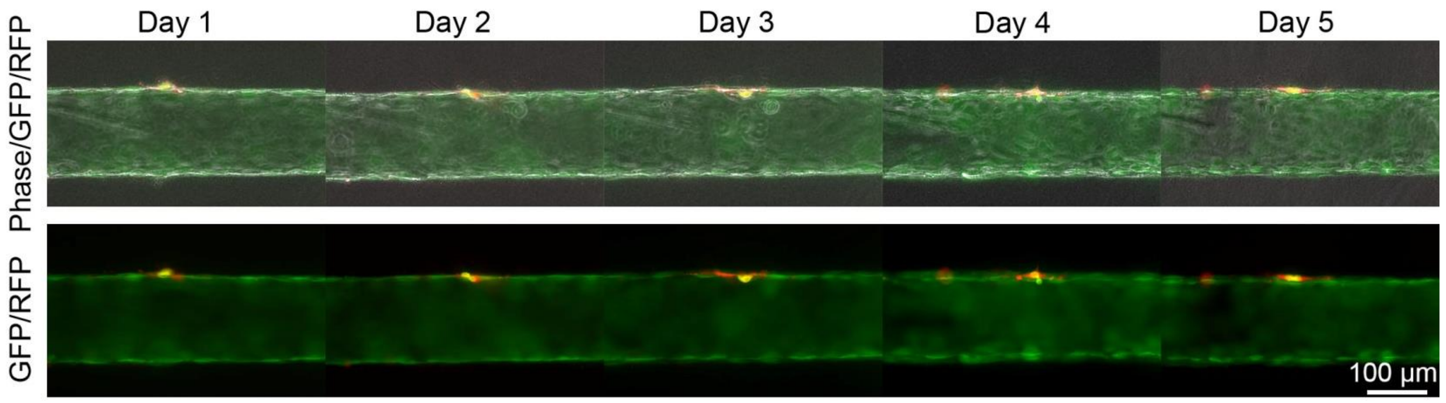
Supplementary Figure S2. Time-series compilation of 8 intravasation events of MDA-MB-231 breast cancer cells (BCCs). **A,** 4 mitosis-mediated intravasation events where cell division is readily identifiable from time-lapse images, resulting in cell rounding, transendothelial migration, and escape into vessel flow. **B,** 4 intravasation events where cell rounding results in transendothelial migration and escape, but mitosis is not conclusively identified. Flow is from left to right. BCCs in events #1-6 were analyzed for circularity and cell width; the results of which are exhibited in Supplementary Figure S4. BCCs in events #7 and #8 could not be analyzed for shape descriptors.



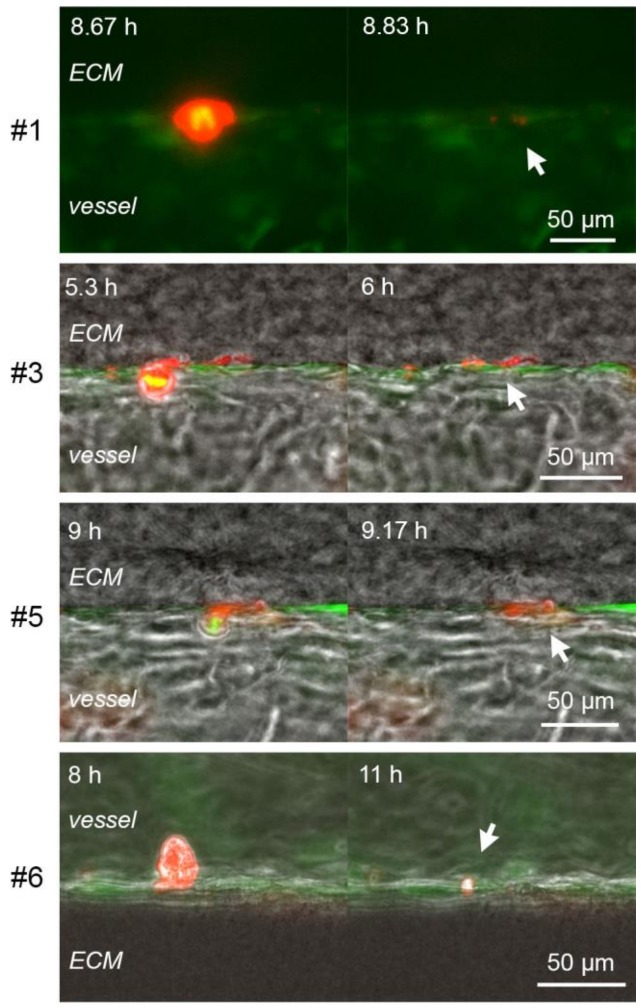
Supplementary Figure S3. Time-series compilation of 3 MDA-MB-231 breast cancer cell (BCC) cell division events at the ECM-vessel interface that did not result in transendothelial migration or intravasation. BCCs in events #1-3 were analyzed for circularity and cell width; the results of which are exhibited in Supplementary Figure S4.



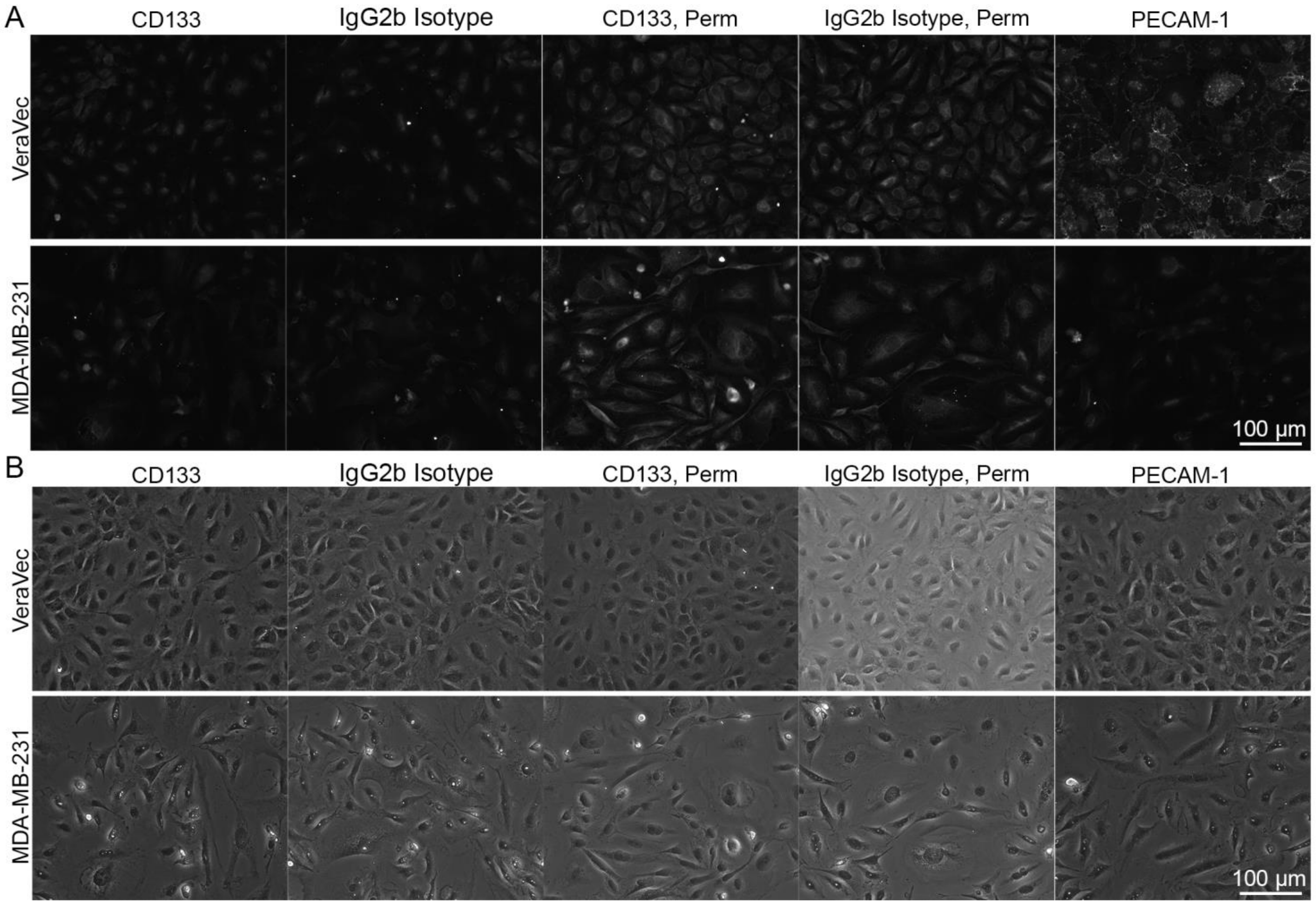
Supplementary Figure S4. Summary of cell shape descriptors vs. time during intravasation and tumor cell division at the ECM-vessel interface. **A,** tumor cell circularity and maximum width vs. time for: (1) mitosis-mediated intravasation, (2) cell rounding and intravasation, and (3) mitosis at the ECM-vessel interface not leading to intravasation. Each graph has the corresponding event number labeled in the upper left corner. Refer to Supplementary Figures S2 and S3 for corresponding time-series images. **B,** time period between the initiation of cell rounding (circularity ≥ 0.6) and cell detachment for each intravasation event (#1-6).



Supplementary Figure S5. Time-series image panel of a single dual-labeled MDA-MB-231 breast cancer cell (BCC) residing at the ECM-vessel interface for 5 days with phase-contrast/GFP/RFP overlay (top) and GFP/RFP overlay (bottom). The vessel endothelium is comprised of VeraVec-GFP cells. The imaging plane is focused at the vessel equator and flow is from left to right. Notice at Day 3, the BCC is exposed to vessel flow after transendothelial migration, but its anchored processes prevent detachment, and it eventually returns to the ECM-vessel interface at Day 5.



Supplementary Figure S6. Time-series images of tumor cell intravasation revealing anchored processes (arrow) remaining at the ECM-vessel interface after tumor cell escape into vessel flow. Flow is from left to right. The RFP fluorescence overlay displays the remnants of dual-labeled MDA-MB-231 breast cancer cell (BCC) cytoplasm in intravasation events #1, 3, 5, 6.



Supplementary Figure S7. CD133 expression on MDA-MB-231 breast cancer cells (BCC) and VeraVec. **A**, Fluorescence images of CD133, PECAM-1, and isotype control immunostaining with and without permeabilization (perm). **B**, Corresponding phase-contrast images. All immunofluorescence images were acquired with the same camera and excitation settings (i.e. exposure time, camera gain, excitation light intensity), and secondary antibody incubation (i.e. donkey anti-mouse IgG conjugated to Alexa Fluor 647). All primary antibodies (i.e. CD133 IgG2b, PECAM-1 IgG1, and Isotype Control IgG2b) are of monoclonal mouse origin. CD133 did not preferentially stain subpopulations of any cell type regardless of permeabilization. PECAM-1 positively stained the membrane of VeraVecs but not that of MDA-MB-231 BCCs.

Supplemental Code

ImageJ Java Plugin to Extract Maximum Cell Width from a Roi Traced Cell

```
/** max_Width plugin instructions
* Copy file into Imagej Plugins folder and 'Compile and Run' in ImageJ
* Program will appear in 'Plugins' dropdown menu after restarting ImageJ
* Run file.  When prompted, open a .zip file containing ImageJ ROI's.
* Program will output the maximum vertical width for each ROI in the
* .zip file.  Output will appear in the ImageJ Results Table.  The max
* width line and upper and lower bounds as points will appear as ROI's
* in the ImageJ ROI Manager for each ROI.
*/

import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.frame.*;
import ij.io.*;
import ij.measure.ResultsTable;
import java.lang.Math;

public class max_Width implements PlugIn {

   public static final boolean DEBUGG = false;
   public void run(String arg) {
      // Filepath and name of file for .zip file containing multiple Roi
      OpenDialog dialog = new OpenDialog("Select ROI File");
      String fpath = dialog.getDirectory();
      String fname = dialog.getFileName();
      String path = dialog.getPath();
      RoiDecoder roiDec = new RoiDecoder(fpath + fname + ".roi");
      ResultsTable rt = new ResultsTable();

      // Close RoiManager if it's already open
      RoiManager roiMan = RoiManager.getInstance();
      if (roiMan != null)
         roiMan.close();

      roiMan = new RoiManager();

      // Opens the .zip file containing multiple Roi
      if (roiMan.runCommand("Open", fpath + fname) == false)
         IJ.showMessage("Roi not openned");

      // Makes sure counter is at least at 1, otherwise it crashes
      if (rt.getCounter() == 0)
         rt.incrementCounter();

      Roi cluster;
      int numRoi = roiMan.getCount(); // Get number of Roi's in RoiManager
      Roi[] widthRoiArray = new Roi[numRoi]; // Stores Roi's of lines
      // Stores Roi's of points
      EllipseRoi[] ellipseRoiArrayUp = new EllipseRoi[numRoi];
```

```
EllipseRoi[] ellipseRoiArrayDown = new EllipseRoi[numRoi];

// Interate through all Roi's in RoiManager
for(int k = 0; k < numRoi; k++){
   cluster = roiMan.getRoi(k);
   PolygonRoi pRoi;
   if (cluster instanceof PolygonRoi){
      pRoi = (PolygonRoi) cluster;
      cluster.setStrokeWidth(2.0);  // Make line width thicker
      int pType = pRoi.getType();
      int numPoints = pRoi.getNCoordinates();
      FloatPolygon allPoints = pRoi.getFloatPolygon();
      float[] Ycoord = allPoints.ypoints;
      float[] Xcoord = allPoints.xpoints;
      // Allocating max space for coordinate arrays
      float[] YcoordD2, YcoordD1 = new float[numPoints];
      float[] XcoordD2, XcoordD1 = new float[numPoints];
      float[] YcoordU2, YcoordU1 = new float[numPoints];
      float[] XcoordU2, XcoordU1 = new float[numPoints];

      // Count variables for up (U) and down (D) roi's
      int countD = 0; int countU = 0;
      float midPoint = middlePoint(Ycoord);
      for (int i = 0; i < numPoints; i++){
         if(Ycoord[i] > midPoint){
            YcoordD1[countD] = Ycoord[i];
            XcoordD1[countD] = Xcoord[i];
            countD++;
         }
         else{
            YcoordU1[countU] = Ycoord[i];
            XcoordU1[countU] = Xcoord[i];
            countU++;
         }
      }

      // Transfer separated points to appropriately sized arrays
      // of floats.
      XcoordD2 = new float[countD];
      YcoordD2 = new float[countD];
      for(int i = 0; i < countD; i++){
         XcoordD2[i] = XcoordD1[i];
         YcoordD2[i] = YcoordD1[i];
      }
      XcoordU2 = new float[countU];
      YcoordU2 = new float[countU];
      for(int i = 0; i < countU; i++){
         XcoordU2[i] = XcoordU1[i];
         YcoordU2[i] = YcoordU1[i];
      }

      // Interate through all upper points and return max width
      float maxWidth = 0;
      float width = 0;
      float widthX= 0;
```

```
        float widthY = 0;
        for (int i = 0; i < XcoordU2.length; i++){
           width = findWidth(XcoordU2[i], YcoordU2[i], XcoordD2,
                    + YcoordD2, rt);
           if (width > maxWidth){
              maxWidth = width;
              widthX = XcoordU2[i];
              widthY = YcoordU2[i];
           }

          if (DEBUGG){
              rt.addValue("width", width);
              rt.addValue("maxWidth", maxWidth);
              rt.incrementCounter();
           }
        }
        //  Add maxWidth and Width starting location in results table
        rt.addValue("maxWidth", maxWidth);
        rt.addValue("X1", widthX);
        rt.addValue("Y1", widthY);
        rt.addValue("X2", widthX);
        rt.addValue("Y2", widthY + maxWidth);

        //  Add widths as lines/rectangular ROIs with 1 px width to
        //  RoiManager
        widthRoiArray[k] = new Roi( (int)widthX, (int)widthY, 1,
           + Math.round(maxWidth) );
        widthRoiArray[k].setStrokeWidth(2.0);
        int elRad = 2;       // Radius of ellipse/sphere
        EllipseRoi ellipseUp = new EllipseRoi((double)widthX+elRad,
           +(double)widthY+elRad,(double)widthX-elRad,
           +(double)widthY-elRad, 1.0);
        EllipseRoi ellipseDown = new EllipseRoi((double)widthX+elRad,
           +(double)widthY+elRad+maxWidth,(double)widthX-elRad,
           +(double)widthY-elRad+maxWidth, 1.0);
        ellipseRoiArrayUp[k] = ellipseUp;
        ellipseRoiArrayDown[k] = ellipseDown;
    }
     if(k < numRoi-1)
        rt.incrementCounter();
} // End for loop iterating through Roi's
for(int k = 0; k < numRoi; k++){
   // Changes image slice to get Roi Slice info into new Roi
   roiMan.select(k);
   roiMan.addRoi(widthRoiArray[k]);
}
for(int k = 0; k < numRoi; k++){
   // Changes image slice to get Roi Slice info into new Roi
   roiMan.select(k);
   roiMan.addRoi(ellipseRoiArrayUp[k]);
}
for(int k = 0; k < numRoi; k++){
   // Changes image slice to get Roi Slice info into new Roi
   roiMan.select(k);
   roiMan.addRoi(ellipseRoiArrayDown[k]);
   }
   rt.show("Results");
}

/** Returns the average as a float between the min and max values
*   in an array of floats
@param yCoord Array of floats
@return The average of the min and max values as a float
*/
public float middlePoint(float[] yCoord){
   float min = yCoord[0];
   float max = yCoord[0];
   for(int i = 0; i < yCoord.length; i++){
      if(yCoord[i] < min)
        min = yCoord[i];
      if(yCoord[i] > max)
         max = yCoord[i];
   }
   return (min + max)/2;
}

/** Finds the width or distance between a point and the opposing set of
   points.  Will interpolate between points if an opposing point
   does not exist.  Assumes a horizontal orientation of the vessel.
   @param x x coordinate of the point in question
   @param y y coordinate of the point in question
   @param xCoord Array of floats of opposing points (xCoordinates)
   @param yCoord Array of floats of oppsing points (yCoordinates)
   @return The distance between the float (x,y) and the analagous point
   in the opposing Arrays
*/
public float findWidth(float x, float y, float[] xCoord,
   +float[] yCoord, ResultsTable rt){
   // Check that we're not on the ends
   float maxX = getMax(xCoord);
   float minX = getMin(xCoord);
   if (DEBUGG){
      rt.addValue("x", x);
      rt.addValue("y", y);
      rt.addValue("maxX", maxX);
      rt.addValue("minX", minX);
   }
   if (x <= minX || x >= maxX){
      return 1;
   }

   // If an opposing x-value exists, return the y difference
   for (int i = 0; i < xCoord.length; i++)
      if (xCoord[i] == x){
         if (DEBUGG)
            rt.addValue("Opp X width", Math.abs(yCoord[i]-y));
         return Math.abs(yCoord[i]-y);
      }

   // If no opposing x-value exists, find x-values that border the
```

```
      // point and interpolate a y value and return the difference
      int leftTrack = 0;
      int rightTrack = 0;
      float xLeft = 0;
      float xRight = 1400;
      // Find left most x-coordinate
      for (int i = 0; i < xCoord.length; i++){
        if(xCoord[i] < x && xCoord[i] >= xLeft){
            xLeft = xCoord[i];
            leftTrack = i;
         }
      }
      // Find right most x-coordinate
      for (int i = 0; i < xCoord.length; i++){
         if (xCoord[i] > x && xCoord[i] <= xRight){
            xRight = xCoord[rightTrack];
            rightTrack = i;
         }
      }
      // Interpolate between the left and right points
      float slope = (yCoord[leftTrack] - yCoord[rightTrack])/
         +(xCoord[leftTrack] - xCoord[rightTrack]);
      float interpolated = yCoord[leftTrack] + (x - xCoord[leftTrack])
         +*slope;
      float width = Math.abs(interpolated - y);
      if (DEBUGG){
         rt.addValue("Int X width", width);
         rt.addValue("yCoord[leftTrack]", yCoord[leftTrack]);
         rt.addValue("yCoord[rightTrack]", yCoord[rightTrack]);
         rt.addValue("xCoord[leftTrack]", xCoord[leftTrack]);
         rt.addValue("xCoord[rightTrack]", xCoord[rightTrack]);
      }
      return width;
   }

   public float getMax(float[] array){
      float max = array[0];
      for (int i = 0; i < array.length; i++)
         if(array[i] > max)
            max = array[i];
      return max;
   }
   public float getMin(float[] array){
      float min = array[0];
      for (int i = 0; i < array.length; i++)
         if(array[i] < min)
            min = array[i];
      return min;
   }
}  // End Class
```