High-Level Sprout Geometry Extraction for In Vitro Angiogenesis Assays

Gio Borje
University of California, Irvine
gborje@uci.edu

Craig Steinke
University of California, Irvine
steinkec@uci.edu

ABSTRACT

We have developed an automated system for the quantitative analysis of *in vitro* angiogenesis assays. Specifically, the system is designed to analyze fibrin gel bead sprouting assays developed by the laboratory of Dr. Chris Hughes in the Department of Molecular Biology & Biochemistry at UC Irvine. Our system enumerates sprouts for each bead and calculates the average branching factor and the average length for an imaged assay. Our approach reports accurate sprout enumeration when compared with manual counts by an expert observer while significantly reducing the analysis time.

1. INTRODUCTION

Angiogenesis is a mechanism for the formation of new blood vessels from pre-existing vessels. Tumor angiogenesis in solid tumors is thought to facilitate the transition from a benign tumor to a malignant phenotype. Additionally, the avascular growth phase allows for an an approximate maximum size of 1–2mm in diameter; on the other hand, the vascular growth phase enables unyielding tumor expansion and metastasis [2]. Subsequently, the analysis of *in vitro* angiogenesis assays is necessary to assess the impact of pro-angiogenic and anti-angiogenic agents.

Niemisto et. al developed a general image analysis method for the quantification of similar *in vitro* angiogenesis assays. Their method provides the length and size of each tubule complex as well as the number of junctions within them [4]. The TCS Cellworks Angiokit (Buckingham, UK) images used by Niemisto et. al have two distinguishable and ideal properties in contrast to our images that require us to use a variant approach. First, their images have tubule complexes that are solidly filled whereas our images have visible lumens. Second, their images show no distinguishable origin for sprouts whereas our images have a bead from which sprouts originate. Hence, we propose a modified methodology for our image set.

The image set under analysis is provided by the Hughes Lab. Their fibrin bead assay using human umbilical vein endothelial cells (HUVEC) contrasts with fibrin bead assays containing Bovine aortic endothelial cells (BAEC). HUVEC have been the canonical EC model system and more accurately represents angiogenesis in humans. The assay begins by culturing HUVEC as a monolayer on dextran-coated Cytodex beads. This method induces HUVEC to recapitulate multicellular capillaries in fibrin gels. Furthermore, the method for HUVEC promotes sprouting, lumen formation and long-term stability of neovessels. The high-resolution images of beads are then captured on an IX70 Olympus microscope [3]. A sample image can be seen on Figure 1.

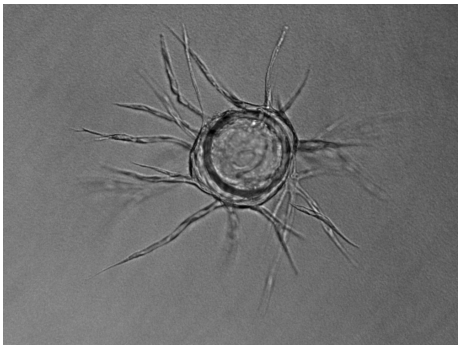


Figure 1: Single Bead Image from the Hughes Lab Images

Previously, the number of sprout counts per bead were measured by manual counting on the obtained images. Furthermore, the sprout length was ambiguously measured in arbitrary units [3]. Our system relieves the laborious, manual analysis by quickly automating the detection and reliably analyzing of imaged assay features.

Our system, the High-Level Sprout Geometry (HLSG) Extractor, is designed to detect features, restore features and analyze the features of imaged assays. Features for detection include the Cytodex bead and its associated multicellular capillaries. Due to the noise and depth of the image, the HLSG Extractor uses structural inpainting to infer which sprout segments emerge belong the same sprout. After structural inpainting, the HLSG Extractor quantitatively analyzed the images using Sholl Analysis to determine the number of primary sprouts, the average length and the average branching factor for each bead in the imaged assay.

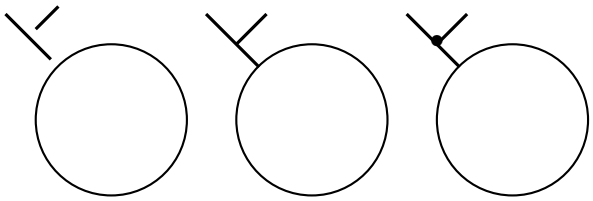
In addition to the High-Level Sprout Geometry (HLSG)

Extractor, a driver and report generator are implemented to drive functionality on sample images and generate reports on the analyses respectively.

The following sections will proceed as follows. Section 2 will describe an overview of the HLSG feature detection, feature restoration and analysis mechanisms. Section 3 will describe the data structures used to represent the HLSG and its properties. Section 4 shows how the system is designed for modularity and robustness.

2. METHODOLOGY

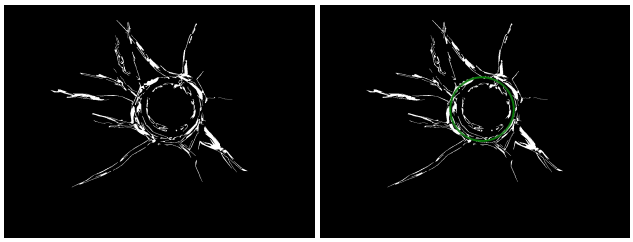
Our system enables feature set detection, minor feature restoration and quantitative analysis which can be decomposed into four stages. The first two stages detect feature sets: beads as features and then sprouts as features. In the third stage, the system attempts to restore a few sprout features by approximating and inpainting connections between broken sprout segments as well as filling holes. Finally, the system quantitatively analyzes the imaged assays through Sholl Analysis.



(a) Sprout Segments (b) Branching (c) Junction Point

Figure 2: Branching Geometry

2.1 Bead Extraction



(a) Edges (b) Bead Detection

Figure 3: Bead Extraction Outline

Bead extraction is a two-step process. To reduce noise, the system first smooths the image using a Gaussian blur. Second, circles in the image are detected using the circular Hough Transform. The circles detected correspond to the beads in the assay. It follows that the origin and radius of the bead is obtained.

In order to avoid false bead detections, we enforce that the minimum distance between the origin of every pair of detected circles be four times the average bead radius i.e. each detected bead should at least be a bead apart. Beware that we ignore the degenerate case in which two or more beads are within the distance constraint because associating sprouts to beads becomes inaccurate.

2.2 Sprout Extraction

(a) Dilated (b) Skeleton

Figure 4: Sprout Extraction Outline

Sprout extraction depends on bead extraction because the beads must be masked before sprout extraction occurs to separate beads from sprouts. We mask the beads given the geometry of the circles. Due to the disconnectivity of sprouts in the assay, we begin by obtaining all sprout segments and represent them as line segments. We distinguish the end points of each line segment for connectivity. The start point, S, is the end point on the line segment such that it is closer to the origin of the line segment's closest bead. The end point, E, is the end point on the line segment such that it is farther from the origin of the line segment's closest bead.

To determine which line segments belong to the same sprout, we use Euclidean distance between their start and end points enforced by the constraint that the end point must be closer to the origin of its closest bead than the target start point. For example, two line segments are part of the same sprout if the distance between the start an end points are within a specified distance parameter, d.

2.3 Sprout Restoration

There are two primary sources of degradation that we have discovered in the imaged assays: disjoint sprout segments and false capillary lumen detections i.e. holes. We will discuss our approaches to both problems independently.

The disjoint sprout segments are caused by the three-dimensional nature of the sprouts moving in and out of focus of the microscope.

False capillary lumen detections are artifacts of the Canny edge-detection algorithm. By the morphology of these structures, we simply find the contours which enclose small holes and perform inpainting by polygon approximation using the Ramer-Douglas-Peucker algorithm.

2.4 Sholl Analysis

Sholl Analysis is a quantitative method for quantitatively analyzing morphological characteristics of neurons [5]. Briefly,

Sholl Analysis consists of counting the number of foreground to background crossings given concentric circles of a parameterized radius.

Algorithm 1 Sholl Analysis

```
procedure SHOLLANALYSIS(img, origin, radius)
    r_min ← radius
    r_max ← min{origin.x, img.width - origin.x}
    r_max ← min{r_max, origin.y, img.height - origin.y}
    crossings ← empty dictionary
    for all r ∈ [r_min, r_max] do
        crossings[r] ← COUNTCROSSINGS(img, origin, r)
    end for
    return crossings
end procedure
procedure COUNTCROSSINGS(img, origin, radius)
    crossings ← 0
    for all pixel ∈ BRESENHAMCIRCLE(origin, radius) do
        if crossing detected at pixel then
            crossings ← crossings + 1
        end if
    end for
    return crossings
end procedure
```

3. DATA STRUCTURES

The following data structures are used to implement the HLSG Extractor.

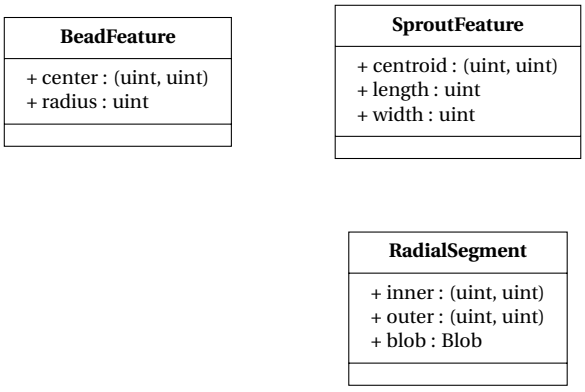


Figure 5: HLSG Extractor Features Class Diagram

3.1 Bead Feature

A bead feature is an abstraction of the Cytodex bead coated with endothelial cells in the assay. The geometry of the bead is intuitively circular; subsequently the geometry can is described by the descriptor in Figure 5.

3.2 Sprout Feature

A sprout feature is an abstraction of the blood vessels that develop through angiogenesis from the designed bead. Subsequently, sprout feature extraction is dependent upon bead descriptors. The sprout is actually comprised of a set of pixel segments because of the possibility that a sprout is disconnected.

3.3 Radial Line Segment

Due to the disconnectivity of sprouts, individual sprout segments are represented by a radially defined line segment. That is, we distinguish the end points from its radial distance from the origin of its corresponding bead. Given a line segment, we say that and end point is the *inner point* if it is radially closer than its complementary end point; otherwise, we call the end point the *outer point*.

In addition to the distinguishable end points, a radial line segment is a line fit onto a corresponding blob of pixels which can be considered a sprout segment.

3.4 Driver

The Driver is responsible for parsing input from the user and emulating the encoded actions as functions of the HLSG Extractor. That is, the Driver acts similar to a REPL (Read-Eval-Print-Loop) that reads input from the user, evaluates the input and prints the corresponding output in a loop. The set of commands available to the user is outlined Table 1.

4. SYSTEM ARCHITECTURE

The system requires Python version 2.7x with the SimpleCV package. The architecture of the system is based on our methodology for quantitatively analyzing *in vitro* angiogenesis. The system, however, incorporates modules for driving batch processes as well as a Read-Eval-Print-Loop (REPL) for console interaction. Finally, a module incorporated for generating CSV reports of the analysis. The sequence diagram for the system components are shown in Figure 6.

The REPL module controls the interaction between the user and the system. Commands available in the REPL are shown in Table 1.

Command	Output	Description
extract [file]	HLSG of file	Extracts the HLSG of the given file.
extract [files]	HLSG of files	Extracts the HLSGs of the given files.
exit	Goodbye	Exits the system.

Table 1: Commands

5. RESULTS

The results of our sprout enumeration show a standard error of 2.5 when compared manual sprout counts of an expert observer.

Display a comparison table with human counts.

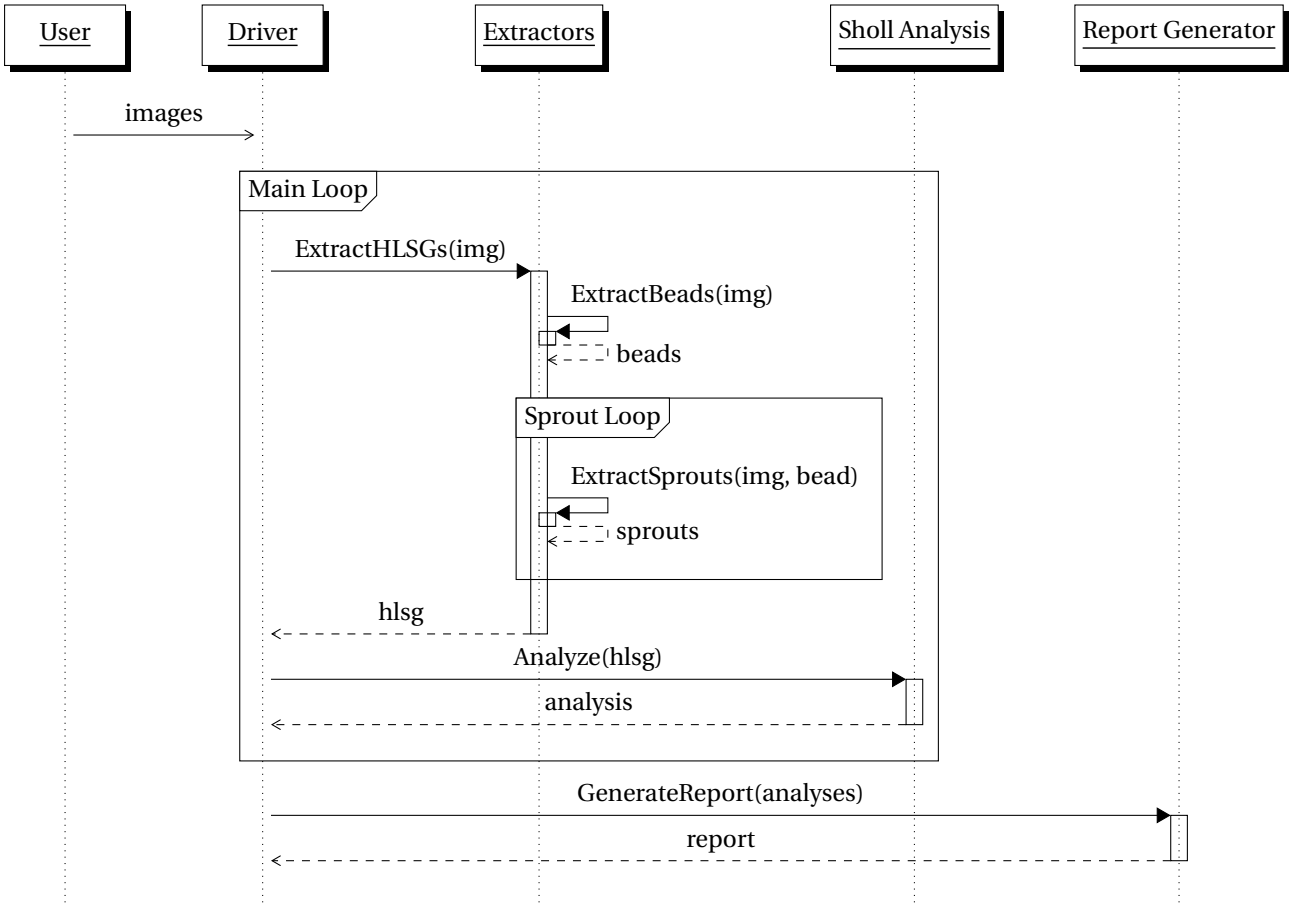


Figure 6: High-Level Architecture

Human Sprout Counts	HLSG Sprout Counts
0	0
0	0
0	0
0	0

Table 2: Result Comparison

6. DISCUSSION

Although the methodology is similar to AngioQuant, our method operates on two-ridge structures [4].

7. REFERENCES

[1] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Commun. ACM*, 20(2):100–106, February 1977.

[2] Robert S. Kerbel. Tumor angiogenesis: past, present and the near future. *Carcinogenesis*, 21(3):505–515, 1999.

[3] Martin N. Nakatsu, Richard C.A. Sainson, Jason N. Aoto, Kevin L. Taylor, Mark Aitkenhead, Sofía Pérez del Pulgar, Philip M. Carpenter, and Christopher C.W. Hughes. Angiogenic sprouting and capillary lumen formation modeled by human umbilical vein endothelial cells (huvec) in fibrin gels: the role of fibroblasts and angiopoietin-1. *Microvascular Research*, 66(2):102 – 112, 2003.

[4] A. Niemisto, V. Dunmire, O. Yli-Harja, Wei Zhang, and I. Shmulevich. Robust quantification of in vitro angiogenesis through image analysis. *Medical Imaging, IEEE Transactions on*, 24(4):549–553, 2005.

[5] D. A. Sholl. Dendritic organization in the neurons of the visual and motor cortices of the cat. *J Anat.*, 87(4):387–406, 1953.

APPENDIX

A. PSEUDO CODE

This section outlines the pseudo-code for the HLSG Extractor operations.

A.0.1 Sprout Extractor

Given an imaged assay and a set of bead features, the algorithm proceeds by masking the beads from the image. Next, a segmentation strategy is used to separate individ-

ual sprouts from the collection of globally detected sprouts. Finally, the segmentation strategy yields the detected feature set of sprouts.

Algorithm 2 Sprout Extraction

```
procedure ExtractSprouts(img, beads)
    maskedImg ← maskBeads(img, beads)
    strategy ← SegmentStrategy(maskedImg,beads)
    sprouts ← strategy.segment()
    return sprouts
end procedure
```

A.0.2 HLSG Extractor

Algorithm 3 HLSG Extraction

```
procedure ExtractHLSGs(img)
    beads ← ExtractBeads(img)
    sprouts ← ExtractSprouts(img, beads)
    hlsgs ← MapSproutsToBeads(sprouts, beads)
    return hlsgs
end procedure
```

A.0.3 Ordered Bresenham Circle Algorithm

Bresenham's circle algorithm is frequently used to determine which pixels should be selected to draw a circle. algorithm utilizes symmetry of a circle in order to reduce the running time by a factor of eight [1]. For our application with Sholl Analysis, the algorithm must return the points as an ordered set which is defined to be counterclockwise starting from the positive x-axis.

The algorithm calculates points for all eight octants simultaneously. The octants are labeled with integers starting with zero. We label the octants from starting from the x-positive axis in the counterclockwise direction. The labels can be seen in figure 7. Because it is necessary that the points are ordered counterclockwise from the positive x-axis, we modify the algorithm. The algorithm calculates the points from ordered clockwise or counterclockwise from the closest coordinate axis. Hence, we reverse the points calculated for every odd octant and then concatenate their data points.

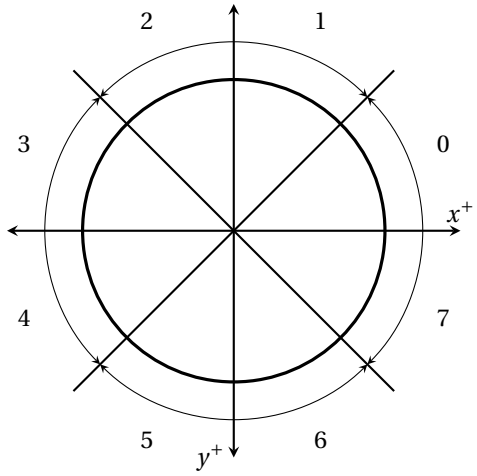


Figure 7: Octants in the Eight-Way Symmetry of a Circle

Algorithm 4 Ordered Bresenham Circle Algorithm

```
Ensure: points returned are ordered counterclockwise
  from the positive x-axis
  procedure BresenhamCircle(origin, radius)
      x ← radius
      y ← 0
      r_error ← 1 − x
      octants ← empty array
      while x ≥ y do
          octants[0] ⇐ (x + origin.x, −y + origin.y)
          octants[1] ⇐ (y + origin.x, −x + origin.y)
          octants[2] ⇐ (−y + origin.x, −x + origin.y)
          octants[3] ⇐ (−x + origin.x, −y + origin.y)
          octants[4] ⇐ (−x + origin.x, y + origin.y)
          octants[5] ⇐ (−y + origin.x, x + origin.y)
          octants[6] ⇐ (y + origin.x, x origin.y)
          octants[7] ⇐ (x + origin.x, y origin.y)
          y ← y + 1
          if r_error < 0 then
              r_error ← r_error + 2y + 1
          else
              x ← x − 1
              r_error ← r_error + 2(y − x + 1)
          end if
      end while
      points ← []
      for all octant ∈ octants do
          if octant is odd then
              reverse the octant
          end if
          for all point ∈ octant do
              points ⇐ point
          end for
      end for
      return points
  end procedure
```