­

Software Requirements Specification­­

for

MicronPro

Version 3.1

Prepared by Jack Kelly

Encompass Technologies

July 7th 2021

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
| MicronLite | 5/20/2020 | Provide simpler output and set requirements for images to pass/fail |  |
| MicronPro | 7/19/2020 | Stack reconfiguration for implementing cloud hosting |  |

# Introduction

## Purpose

This document outlines in detail the Software *MicronPro* v3.1. This document aims to provide details into how the software is constructed, the goal of the software, and proper use of the software. In addition this document goes into detail about the construction of the software, and the methods used to compute various important values generated by the program

## Document Conventions

**Terminology:**

**Job:** this refers to a frame, or batch of frames to be processed or that have been processed

**Frame:** when used, refers to a set of images

**Worker:** a “worker” is a computer dedicated to capturing and processing the images.

## Intended Audience and Reading Suggestions

This document is intended for project managers lab technicians or anyone else who may interact with the program or the programs results. After reading, the user should understand use-cases and functionality of the program at a high level. In addition, this document provides information about how the program processes images, which is critical to understand in order to properly use the tool. Finally, an in-depth overview of the system outlines the relationships between object-classes and modules included in the program.

## Product Scope

MicronPro applies image processing to microscopic images of implantable Cardiovascular filters measuring the overall porosity of the filter and in addition find pores with sizes that exceed the specified requirements and outputs the results to an excel spreadsheet to be reviewed by a technician. The system can be broken down into three parts, website, Heroku cloud server and a “worker”. All of these parts and their interactions will be detailed in this document

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

ion.>

# Overall Description

## Product Perspective

This development of this software began in June of 2020. The program was initially created as an alternative to manually using MiPar(a desktop image processing program) to find the porosity of the filter. While MiPar is a powerful tool, more information and a streamlined and customized output was required, leading to the development of MicronLite. The program has now evolved to provide more information on the filter, such as finding the largest pores and the size of the largest circle that can be inscribed in the pore. The output of the program is also formatted into a spreadsheet which provides a concise and detailed report of data gathered from the images.

## Product Functions

* Allows users to login to the system from a web browser
* Allows users to create “jobs” to run on worker computers
* Allows users to save and load job configurations
* Allows users to view completed and in progress “jobs”
* Allows users to verify that images have been reviewed to update the status to “verified”
* Store job data redundantly on local computer and cloud
* Input single/folder of microscopic images of the device
* Find the largest pores and the size of the largest particle that could pass through
* Find the porosity (portion of the image not filled by the fibers)
* Output simple details about images that ‘passed’ the requirements
* Output more detailed information about images that did not ‘pass’ the requirements

## User Classes and Characteristics

**Software Admin:** responsible for understanding all of operations of the program, the architecture of the program, how the software will be implemented and performing updates to the program as necessary, administrator access to the computer running the program is required.

**General User:** a general user of the program is someone who is responsible for running images through the software. This user should not adjust any of the parameters that the program uses unless directed by a Technical user

**Technical User:** This user should have a solid grasp on how the program extracts the values from the image and understand how changing the threshold value impacts the results. This user may also be responsible for capturing the images the program will run should have an idea knowing what images are suitable for the program (*see section 2.5)*.

## Operating Environment

The software is designed to function both on Windows and Unix based computers but is planned to only be used in a computer with an Enterprise Windows 10 environment. Since the program implements specific versions common python libraries, a python virtual environment will be used to ensure that dependencies are isolated independent of the systems python environment which could be subject to change overtime by other applications. Also important for windows machine is having visual studio build tools installed (link is in readme.md). This installs C dependencies that your windows machine may or may not have already installed. This step may not be necessary for future versions.

## Diagram Description automatically generatedDesign and Implementation

## Overview:

## The figure above outlines the three distinct parts of the application and their interactions with the database. Simply put, the dashboard interacts with a backend server (Flask App) that dispatches requests that trigger jobs on the worker computer (running MicronPro Worker).

## External Services *Implemented*

## Heroku: Heroku is a cloud application platform that hosts both the website and the backend server for this system. The platform is linked to the git repositories so as branches are merged into the production version the program will be rebuilt and deploy the newest version of the Flask App.

**MongoDB:** This is a popular cloud document-based database. Mongo is used to keep a record of previous jobs, this record does not include the processed images but includes important details about passing and failing images along with a reference to the created spreadsheet. Also stored here are users consisting of a username and hashed password and stats about the application.

***System Components***

**Flask App(backend):** This portion is responsible for triggering jobs for workers, updating stats and editing jobs. The backend also provides the authentication for the frontend application by using JSON Web Tokens to authenticate users before they are able to interact with the API. After a user has been authenticated the backend will pull data from Mongo about recent jobs and relay it to the frontend to be displayed.

**MicronPro Dashboard(frontend):** This is the website users will sign-in to in order to queue new jobs. After signed in the user will be allowed to view available workers and their folders set a configuration and start a job. For completed jobs the user can navigate to the previous jobs page to view job data. By selecting review image on a job the user is able to set images as reviewed, after reviewing the user can update the changes with the update status button on the bottom of the component.

**MicronPro-Worker:** The bulk of the programs’ logic rests in this component. This component is installed on the computer that will be storing images. This component runs a light flask server only accessible by the Heroku backend to lower exposure. After a job is triggered from the web app a post request will be sent to the worker. This kicks off the execution of the job and creation of the spreadsheet. Once finished the job document is updated in the MongoDB and the output files are stored in a folder synced with OneDrive.

## Design and Implementation

**Identifying pores:**

During the process of analyzing the image the image goes through a filtering process called thresholding which when given a pixel value, separates the image in to background and foreground, creating a binary image. From here, the foreground is separated into disjoint regions, where each region is a pore in the image.

**Calculating porosity**

After the pores have been identified and the image is separated into regions. Determining the porosity is done by adding the sum of the areas of the regions then dividing by the total area

**Identifying Largest diameter pores:**

After calculating the porosity and segmenting the regions, the process of finding the pores with the largest diameter begins. This is one of the most, if not the most computationally taxing processes of the program. Although, at this point we know which pores are the largest but in order to find the largest inscribed circle the pore. To do this the program tests circles of decreasing size in each region until a circle is found. A proof of the algorithm used here will be provided in this document.

**Image sizing – constraint**

The most significant implementation constraint is the resolution of the pictures. Currently the picture size is 600x800 pixels. At this magnification the scale of microns/pixel is 2.49 microns per pixel. This is relevant because it means that a one pixel error would make the measurement off by 2.49 microns. This is also important to know because diameter measurements will also be increments of 2.49, this is why reported areas and diameters will seem clumped around certain

values.

## User Documentation

User documentation can be found in each of the git repositories in the readme.md files. These files explain installation and setting configuration variables. There is also a brief troubleshooting section in these files that contains common fixes to installation errors.

**Git Repos**

MicronPro-worker: <https://github.com/Jack-kelly-22/micronPro-worker>

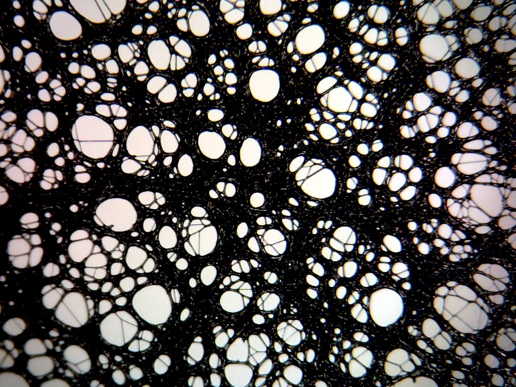
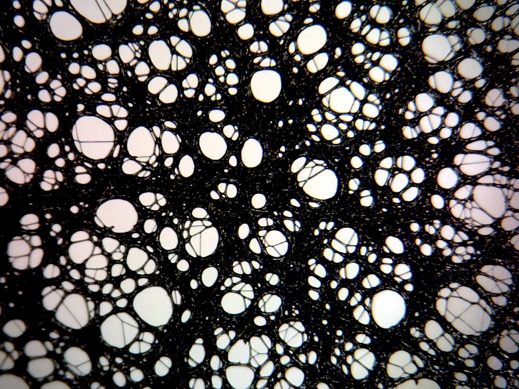
MicronPro-heroku: <https://github.com/Jack-kelly-22/micronPro>

MicronPro-frontend: <https://github.com/Jack-kelly-22/Micron-pro-frontend>

## Assumptions and Dependencies

**Image Quality:**

The consistency and quality of images of critical importance. The program uses adaptive thresholding to identify what is fiber, and what is void space. The adaptive thresholding relies on the contrast between pixels making it important that images are as in-focus as possible. It seems common that the nature of the fixture and shape result in images where the filter is slightly out of focus. To solve this problem the user can set a border of size *x* to ignore outermost x/2(must set even border size. pixels along each edge. Images that are more uniform or at least symmetrical with lighting are ideal

**** **Desirable Images:**

A picture containing outdoor, bread

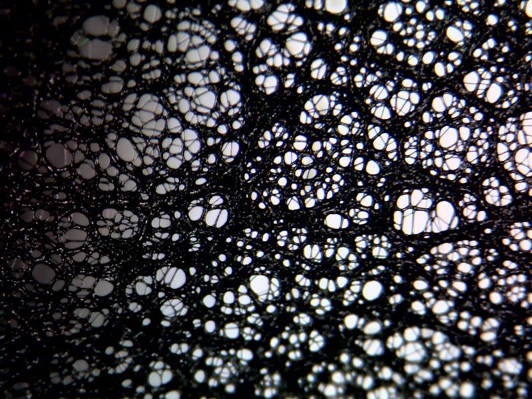
Description automatically generated ***Flawed Images:***

Figure 2: flawed image(out of focus)

Figure : flawed image 1(inconsistent lighting)

A picture containing graphical user interface

Description automatically generated

# External Interface Requirements

## User Interfaces

**User Interface(worker computer):** This is the interface that the user will interact with to process images. The user can select a folder to view from the button with the label “View Folder”, from here the user selects the folder or folders of images that they would like to execute the program on. On the right-hand side of the interface are options that modify how the program will execute, including the scale factor, thresholding value and the pass/fail requirements for the image. To run the program the user simply clicks the run button once the images are selected. The interface (Figure 1) is built ‘TkInter’ which is default python package for creating user interfaces. This process will also be explained in the User usage document

Graphical user interface, application, email

Description automatically generated

Figure – User interface(Tkinter)

**Excel Report:** This is the where the data gathered by the program is output. The spreadsheet is divided into four main parts: Header, Fixture Reference Settings, Inspection Specifications, Cell Inspection Results and Rejected Images.

**Header:** Includes Lot #, Revision #, and the date

Table

Description automatically generatedTable

Description automatically generated **Fixture Reference Settings:** These values reflect the user input parameters that the output was calculated with. These are the same options as shown on the right side of the primary user interface.

**Inspection Specifications:**  contained in this section is the Maximum pore size and Minimum Pore size that are required for the image to be classified as “passed”. If either of these values defy these parameters more information will be printed about the image in the Rejected Images” section.

Table, timeline

Description automatically generated with medium confidence

**Inspection overview:** lists how many pores the program identified over the min\_ignore size, number of pores found over the max allowed size.

Table

Description automatically generated **Cell Inspection Results:**  This section contains a line for each image reporting image name, observed porosity, maximum pore size, and if the image passed or failed.

**Rejected Images**: This section will only contain output for images that fail. In the case of a failed image the program will give the diameter(microns) of the pore/s that caused the image to fail along with the circles center in terms of pixels. The image will only be colored if debug=True

A picture containing text

Description automatically generated

## Hardware Interfaces

Images for the program are taken on a <camera name>. The images are taken at a zoom of <insert zoom> at a distance <insert distance> away from the filter. The Camera is plugged in via USB and stores the images on the computer, to be input to the program later. To use this program the user will need a keyboard and mouse.

## Software Interfaces

**External Python Dependencies:**

1. Pandas:

Version:

Description: data science/analysis package used to store multi-dimensional arrays. One of the most common packages to work with large amounts of data.

Source: <https://pandas.pydata.org/>

1. Numpy:

Version: v 1.20.0

Description: package for working with multi-dimensional arrays. Used in *MicronLite* to store and apply processes to images. This package is widely used for creating scientific and statistical models.

Source: <https://numpy.org/>

1. openPyxl:

Version: v3.0.7

Description: provides an easy method to create spreadsheets in python. Used to format and fill the output excel file.

Source: <https://openpyxl.readthedocs.io/en/stable/>

**Internal Python Dependencies:**

*Note these are included installation of python*

1. os module:

Description: used to interact with system paths to open, read and store data.

Source: <https://docs.python.org/3/library/os.html>

1. Sqlite3

Description: used to store image data locally for reference later if needed

Source: <https://docs.python.org/3/library/sqlite3.html>

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

**Development and use:**

Computer that it is installed on uses RDP to provide Users, Project Managers and the Software Admin access via Microsoft Remote Desktop. In the future the goal is for this program to be ran on an instance of a cloud based virtual machine.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

**Data Schema/Model:**

1. **Entity sets:**

* **Job(at creation):**
  + - Thresh (pixel value that separates foreground and background)
    - min\_porosity (as fraction of 1)
    - max\_porosity
    - min\_ignore (size in microns of area sizes to ignore)
    - max\_allowed (size in microns of the max allowed pore diameter)
    - scale(microns/px)
    - border(x where x is pixels creating padding x/2 padding along each edge)
    - Notes
    - Debug(if true colors out images)
* **Job(at completion):**
  + - num\_pores (total pores found over ignore size)
    - num\_pores\_failed(number of pores over max\_allowed)
    - avg\_porosity
    - num\_images
    - frame\_ls (list of **Frame** entities in job)
    - flagged(default false, can be set to true from previous jobs page)
    - …Job(at creation)
* **User:**
  + - **user\_name**
    - **Pass\_hash**
* **Image:**
  + - Img\_name
    - Violated\_Circles(list of circles failing spec)
    - Pass(Boolean, did meet spec?)
    - Porosity
* **Frame:**
  + - Image\_data(list of **Image** entities)
    - Violated\_Circles(list of circles failing spec)
    - Pass(Boolean, did meet spec?)
    - Failed\_Images (list of image names that failed)

## Measure Image porosity

Description and Priority:

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

This feature is to calculate the porosity, which is the fraction of void space of the material in the image. In the case of this program the images consist of web of fibers, where the void space is the area not covered by the fibers. The void space is illuminated by the fixture creating white pixels while fibers appear black.

### Stimulus/Response Sequences

This is feature is used for each image defined in the Job. The user can set the min and max allowed porosity. If this measurement outside of this range it will “fail” the image.

### Functional Requirements

The porosity

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Segment images (and so on)

# Verification and Validation

## Continuous Integration

The MicronPro-worker git repository is linked to TravisCI which builds the program and then executes a suite of tests that verify the requirements were installed and then runs a series of test cases using Pytest(will be adding more content here).

## Software standards

* Github for branch/version control
* enforces standard python “black” formatting

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>