ISSN: 2639-4383

Annals of Cardiology and Vascular Medicine

Open Access | Research Article

Integrated Program for Optimizing Analysis of Fluorescence Immunostaining in Vascular Smooth Muscle Cells: A Time-Efficient and Bias-Reduced Approach

Nicole Lalime[1]; Ka-Yi Chan[1]; Hongyu Qiu[1-3]*

[1]Translational Cardiovascular Research Center, College of Medicine-Phoenix, University of Arizona, USA.

[2]Department of Internal Medicine, College of Medicine-Phoenix, University of Arizona, USA.

[3]Clinical Translational Sciences (CTS) and Bio5 Institution, University of Arizona, USA.

***Corresponding Author(s): Hongyu Qiu**

Translational Cardiovascular Research Center, College of Medicine-Phoenix, University of Arizona, 475 N 5th Street, BSPB E818, Phoenix, AZ, 85203, USA.

Tel: 602-827-2817; Email: hqiu@arizona.edu

Received: July 14, 2025

Accepted: Aug 12, 2025

Published Online: Aug 19, 2025

Journal: Annals of Cardiology and Vascular Medicine

Publisher: MedDocs Publishers LLC

Online edition: http://meddocsonline.org/

Copyright: © Qiu H (2025). *This Article is distributed under the terms of Creative Commons Attribution 4.0 International License*

Keywords: Fluorescence analysis; Immunostaining; Vascular smooth muscle cells; Semi-automated triangle program; Protein expression.

Abstract

Background: Quantification of protein expression in cardiovascular tissues and cells is critical for understanding disease mechanisms and developing effective therapeutics. However, manual bioimage analysis using platforms like Fiji/ImageJ is often time-consuming, user-dependent, and prone to variability.

Objective: To address these limitations, we developed a semi-automated Jython-based script within Fiji to streamline the analysis of immunofluorescence staining in isolated mouse Vascular Smooth Muscle Cells (VSMCs).

Methods: The script integrates built-in Fiji commands, plugins, and user-defined parameters to automate key steps, including channel separation, thresholding, particle analysis, and data export. The program incorporates a hybrid thresholding approach that combines the automated Triangle method with a user-adjustable green adjustment percentage to improve segmentation accuracy. A fixed red threshold percentage supports consistent quantification of protein markers localized within the mask overlaying the VSMCs.

Results: Compared to manual analysis, this integrated method significantly reduced processing time by 98%, minimized overestimation of cell area, and maintained consistency in intensity measurements.

Conclusion: This flexible and user-friendly workflow enhanced efficiency and reproducibility, offering potential for broader adaptive and versatile analyses across different cell types and experimental conditions in cardiovascular research.

Cite this article: Lalime N, Chan KY, Qiu H. Integrated Program for Optimizing Analysis of Fluorescence Immunostaining in Vascular Smooth Muscle Cells: A Time-Efficient and Bias-Reduced Approach. Ann Cardiol Vasc Med. 2025; 8(1): 1089.

Introduction

The identification and quantification of interested proteins in cardiovascular tissues and cells are essential for understanding disease mechanisms and evaluating treatment efficacy [1]. Immunofluorescence staining is a widely used technique for detecting and localizing specific proteins involved in cardiovascular diseases, both in clinical diagnostics and basic science research [2,3]. Following staining, samples are typically imaged using fluorescence microscopy and analyzed to quantify protein expression [4].

ImageJ has been one of the most widely adopted platforms for bioimage analysis since its release in 1997 due to its open-source nature, broad functionality, and accessibility [5]. Its user-friendly graphical interface allows researchers to perform a wide range of image processing tasks, including manual segmentation, quantification, and measurements. However, as advances in imaging technologies generate increasingly large and complex datasets, manual image analysis becomes time-consuming, labor-intensive, and potentially error-prone [6]. Furthermore, traditional manual processing and a lack of standardization can introduce variability between users, increase susceptibility to bias, and reduce sensitivity, ultimately compromising the accuracy and reproducibility of quantitative results [7,8].

These challenges are particularly pronounced when analyzing primary cells, such as Vascular Smooth Muscle Cells (VSMCs), due to their considerable inherent heterogeneity in shape and size, as well as the potential morphological changes in response to cultural conditions or treatment interventions [9,10]. To address these concerns, researchers have increasingly turned to Fiji, an enhanced distribution of ImageJ that offers an extensive plugin library and supports the development of scripts [11]. Despite Fiji's powerful capabilities, the sheer variety of available tools can make it difficult to identify or develop task-specific scripts appropriate for applications such as cell quantification, viability assessment, or motility analysis [12-15].

Given the growing significance of bioimage analysis in biomedical research, we aimed to develop a high-efficiency, user-friendly, and effective script tailored to quantify protein expression in cultured primary mouse VSMCs. We developed a Jython-based program that integrates built-in Fiji commands and plugins to automate the identification and quantification of immunofluorescence-labeled proteins across multiple samples. Our approach enhances efficiency, ensures consistent analysis, and minimizes unintentional user bias when analyzing large imaging datasets. This work demonstrates a practical and adjustable method with strong potential for broad application in cardiovascular studies, contributing to the ongoing effort to improve reproducibility and throughput in image-based analysis.

Methods

Cell isolation, culture, and immunostaining

Primary VSMCs were isolated from the aortas of Wild-Type (WT) mice and cultured in VSMC medium. Cultured Cells at passage 3 were fixed with 4% paraformaldehyde and then stained with specific antibodies against two independent proteins, e.g., α-Smooth Muscle Actin (α-SMA) and Interleukin-36 receptor (IL-36R). Nuclear staining was performed using DAPI (4',6-diamidino-2-phenylindole).

Image acquisition

VSMCs stained for IL-36R, α-SMA, and nuclei were imaged using a Leica Thunder Wide Field Fluorescence Microscope. Fluorescent signals for the nucleus, α-SMA, and IL-36R were captured in the blue, green, and red channels, respectively. Each image set was saved as a series within a project file and exported in Leica Image Format (LIF) for subsequent analysis.

Set up and imports

The custom image analysis script was developed in Jython (version 2.7.4) and executed within Fiji (ImageJ version 1.54p), an open-source platform that includes a wide range of plugins and supports scripting. The script utilizes core Fiji/ImageJ libraries to perform image processing and analysis tasks [16,17].

Specifically, the script imports IJ and WindowManager for general image operations and window management; Results Table and Measurements to handle measurement data and set analysis parameters; and ParticleAnalyzer and Analyzer to detect objects and perform statistical calculations. Additionally, RoiManager and Channels are used for managing regions of interest and separating image channels, while Overlay and FileSaver facilitate graphical annotations and the saving of processed images. To support microscopy-specific file formats such as Leica Image Format (.lif), the script integrates Bio-Formats libraries. The script also employs Java standard libraries as well as Python standard modules.

User-specific parameters

The user must define two file paths for analysis and data export, as well as set four user-specific parameters before running the script. This files path specifying the data export location hard-coded in the script to avoid repetitive input between multiple sample sets. Subsequent paths for the processed images and measurement results will beproduced automatically throughout the analysis. Upon execution, the user is prompted to enter the four parameters along with the path to the folder containing the LIF files. These parameters include the file name, green adjustment percentage, red threshold percentage, and minimum particle size. All prompts must be completed for the script to proceed. Failure to do so results in an error message. While users can adjust any parameter for each dataset, the program is designed with the intention that the red threshold percentage and minimum particle size remain constant throughout the analysis. For the current study, the red threshold percentage and minimum particle size are set to 7.5 and 500, respectively.

Intensity thresholds

Determining the threshold for the green channel is a critical step in bioimage analysis, as α-SMA is used to identify the VSMCs. The threshold is initially set automatically using the pre-established Triangle plugin [18]. Although the Triangle method enables consistent, user-independent segmentation, the method is a better fit for images containing a high number of cells and a skewed intensity histogram. Consequently, the method may fail to include cells or partial cell regions exhibiting low fluorescence intensity, leading to incomplete cell masks and reduced quantification accuracy.

To address these limitations, our method incorporates a user-adjustable Green Adjustment Percentage (GAP) along with an optional testing phase, allowing the minimum green threshold to be fine-tuned beyond the purely mathematical threshold (Eq. 1). This adjustment can be either positive or negative to increase or decrease the mask size, respectively. The calculation is performed using the following variables: GMV (green minimum value), TMV (Triangle Minimum Value), and GAP (green adjustment percentage expressed as a decimal).

$$GMV = TMV - (TMV*GAP) \qquad \text{(Eq. 1)}$$

The red channel threshold is a user-defined parameter that sets an integer value relative to the green threshold, typically as a positive percentage. A positive value lowers the red minimum threshold slightly below the green threshold, ensuring that the red channel mask is fully contained within the green channel mask during particle analysis (Eq. 2). The script automatically converts these percentage values to decimals. The value is also intended to be constant throughout the analysis. The calculation is performed using the following variables: RMV (red minimum variable), GMV (green minimum value), and RTP (red threshold percentage expressed as a decimal).

$$RMV = GMV*(1-RTP) \qquad \text{(Eq. 2)}$$

Saving original images and testing the green adjustment percentage

Before beginning the analysis, the user has the option to save unprocessed green and red channel images as JPGs and to test the green adjustment percentage. The testing phase begins with the first series in the LIF file, but the user can specify the number of series tested. After completing a test cycle, the user is prompted to either repeat the process with an optional new adjustment value or proceed. The iterative testing continues until the user confirms completion. Once testing is complete, the user is given a final opportunity to revise the green adjustment percentage before the primary analysis begins.

Set measurements and particle analysis

The script generates a continuous Results Table and Summary Table for each series. The red channel will be redirected onto the green channel for analysis. The Results Table records the following parameters: series label, area, mean grey value (intensity), standard deviation, mode, median, integrated density, and raw integrated density. The Summary Table compiles overall metrics and includes the following: series label, total cell count, total area, average particle size, percent area, mean grey value (intensity), mode, integrated density, median, series number, and the minimum intensity thresholds applied to the green and red channels. The Summary Table must be closed manually between each dataset. During particle analysis, the program is configured to include holes within each Region of Interest (ROI). While not applied in the present study, the script also supports the exclusion of edge particles.

Data export

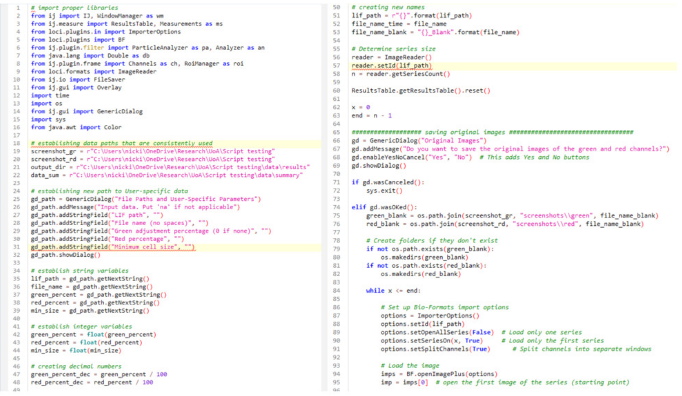
The script is designed to export four output file types. For each series being processed during the analysis, JPG images of the processed red and green channels are saved. The green channel image includes a red threshold mask, yellow outlines indicating detected particles, and numbered labels corresponding to entries in the Results Table. The red channel image includes only the red threshold mask without outlines or labels. Additionally, the cumulative Results Table and Summary Table are separately exported as comma-separated values files for further analysis.

Statistics

Data were presented as average±standard deviation. Statistical analysis was done using unpaired t-tests or a standard one-way ANOVA followed by Tukey-Kramer multiple comparisons. Statistical significance was established at a *p*-value less than 0.05 ($p<0.05$) or a q-value greater than 4.030 ($q>4.030$).

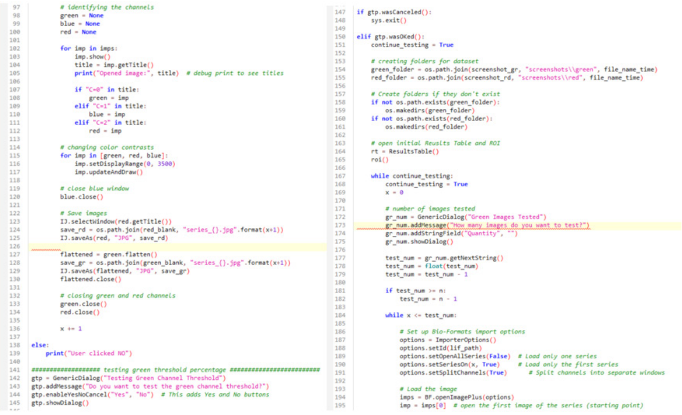
Proposed script

The script used for the data analysis is presented below.



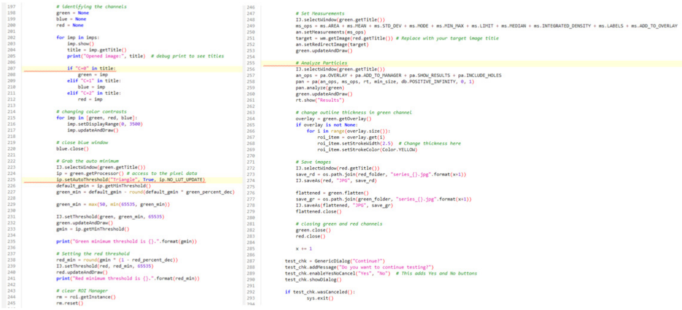
```python
# import proper libraries
from ij import IJ, WindowManager as wm
from ij.measure import ResultsTable, Measurements as ms
from loci.plugins.in import ImporterOptions
from loci.plugins import BF
from ij.plugin.filter import ParticleAnalyzer as pa, Analyzer as an
from java.lang import Double as db
from ij.plugin.frame import Channels as ch, RoiManager as roi
from loci.formats import ImageReader
from ij.io import FileSaver
from ij.gui import Overlay
import time
import os
from ij.gui import GenericDialog
import sys
from java.awt import Color

# establishing data paths that are consistently used
screenshot_gr = r"C:\Users\nicki\OneDrive\Research\UoA\Script testing"
screenshot_rd = r"C:\Users\nicki\OneDrive\Research\UoA\Script testing"
output_dir = r"C:\Users\nicki\OneDrive\Research\UoA\Script testing\data\results"
data_sum = r"C:\Users\nicki\OneDrive\Research\UoA\Script testing\data\summary"

# establishing new path to User-specific data
gd_path = GenericDialog("File Paths and User-Specific Parameters")
gd_path.addMessage("Input data. Put 'na' if not applicable")
gd_path.addStringField("LIF path", "")
gd_path.addStringField("File name (no spaces)", "")
gd_path.addStringField("Green adjustment percentage (0 if none)", "")
gd_path.addStringField("Red percentage", "")
gd_path.addStringField("Minimum cell size", "")
gd_path.showDialog()

# establish string variables
lif_path = gd_path.getNextString()
file_name = gd_path.getNextString()
green_percent = gd_path.getNextString()
red_percent = gd_path.getNextString()
min_size = gd_path.getNextString()

# establish integer variables
green_percent = float(green_percent)
red_percent = float(red_percent)
min_size = float(min_size)

# creating decimal numbers
green_percent_dec = green_percent / 100
red_percent_dec = red_percent / 100
```

```python
# creating new names
lif_path = r"{}".format(lif_path)
file_name_time = file_name
file_name_blank = "{}_Blank".format(file_name)

# Determine series size
reader = ImageReader()
reader.setId(lif_path)
n = reader.getSeriesCount()

ResultsTable.getResultsTable().reset()

x = 0
end = n - 1

################### saving original images ###################################
gd = GenericDialog("Original Images")
gd.addMessage("Do you want to save the original images of the green and red channels?")
gd.enableYesNoCancel("Yes", "No")  # This adds Yes and No buttons
gd.showDialog()

if gd.wasCanceled():
    sys.exit()

elif gd.wasOKed():
    green_blank = os.path.join(screenshot_gr, "screenshots\\green", file_name_blank)
    red_blank = os.path.join(screenshot_rd, "screenshots\\red", file_name_blank)

    # Create folders if they don't exist
    if not os.path.exists(green_blank):
        os.makedirs(green_blank)
    if not os.path.exists(red_blank):
        os.makedirs(red_blank)

    while x <= end:

        # Set up Bio-Formats import options
        options = ImporterOptions()
        options.setId(lif_path)
        options.setOpenAllSeries(False)  # Load only one series
        options.setSeriesOn(x, True)     # Load only the first series
        options.setSplitChannels(True)      # Split channels into separate windows

        # Load the image
        imps = BF.openImagePlus(options)
        imp = imps[0]  # open the first image of the series (starting point)
```



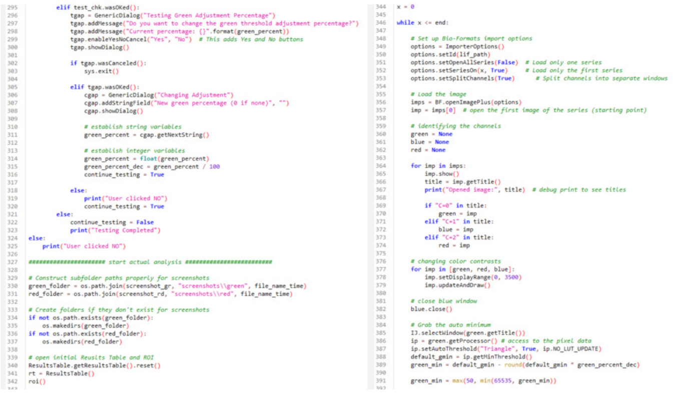
```python
        # identifying the channels
        green = None
        blue = None
        red = None

        for imp in imps:
            imp.show()
            title = imp.getTitle()
            print("Opened image:", title)  # debug print to see titles

            if "C=0" in title:
                green = imp
            elif "C=1" in title:
                blue = imp
            elif "C=2" in title:
                red = imp

        # changing color contrasts
        for imp in [green, red, blue]:
            imp.setDisplayRange(0, 3500)
            imp.updateAndDraw()

        # close blue window
        blue.close()

        # Save images
        IJ.selectWindow(red.getTitle())
        save_rd = os.path.join(red_blank, "series_{}.jpg".format(x+1))
        IJ.saveAs(red, "JPG", save_rd)

        flattened = green.flatten()
        save_gr = os.path.join(green_blank, "series_{}.jpg".format(x+1))
        IJ.saveAs(flattened, "JPG", save_gr)
        flattened.close()

        # closing green and red channels
        green.close()
        red.close()

        x += 1

else:
    print("User clicked NO")

################### testing green threshold percentage ########################
gtp = GenericDialog("Testing Green Channel Threshold")
gtp.addMessage("Do you want to test the green channel threshold?")
gtp.enableYesNoCancel("Yes", "No")  # This adds Yes and No buttons
gtp.showDialog()
```

```python
if gtp.wasCanceled():
    sys.exit()

elif gtp.wasOKed():
    continue_testing = True

    # creating folders for dataset
    green_folder = os.path.join(screenshot_gr, "screenshots\\green", file_name_time)
    red_folder = os.path.join(screenshot_rd, "screenshots\\red", file_name_time)

    # Create folders if they don't exist
    if not os.path.exists(green_folder):
        os.makedirs(green_folder)
    if not os.path.exists(red_folder):
        os.makedirs(red_folder)

    # open initial Reuslts Table and ROI
    rt = ResultsTable()
    roi()

    while continue_testing:
        continue_testing = True
        x = 0

        # number of images tested
        gr_num = GenericDialog("Green Images Tested")
        gr_num.addMessage("How many images do you want to test?")
        gr_num.addStringField("Quantity", "")
        gr_num.showDialog()

        test_num = gr_num.getNextString()
        test_num = float(test_num)
        test_num = test_num - 1

        if test_num >= n:
            test_num = n - 1

        while x <= test_num:

            # Set up Bio-Formats import options
            options = ImporterOptions()
            options.setId(lif_path)
            options.setOpenAllSeries(False)  # Load only one series
            options.setSeriesOn(x, True)     # Load only the first series
            options.setSplitChannels(True)      # Split channels into separate windows

            # Load the image
            imps = BF.openImagePlus(options)
            imp = imps[0]  # open the first image of the series (starting point)
```



```python
            # identifying the channels
            green = None
            blue = None
            red = None

            for imp in imps:
                imp.show()
                title = imp.getTitle()
                print("Opened image:", title)  # debug print to see titles

                if "C=0" in title:
                    green = imp
                elif "C=1" in title:
                    blue = imp
                elif "C=2" in title:
                    red = imp

            # changing color contrasts
            for imp in [green, red, blue]:
                imp.setDisplayRange(0, 3500)
                imp.updateAndDraw()

            # close blue window
            blue.close()

            # Grab the auto minimum
            IJ.selectWindow(green.getTitle())
            ip = green.getProcessor() # access to the pixel data
            ip.setAutoThreshold("Triangle", True, ip.NO_LUT_UPDATE)
            default_gmin = ip.getMinThreshold()
            green_min = default_gmin - round(default_gmin * green_percent_dec)

            green_min = max(50, min(65535, green_min))

            IJ.setThreshold(green, green_min, 65535)
            green.updateAndDraw()
            gmin = ip.getMinThreshold()

            print("Green minimum threshold is {}.".format(gmin))

            # Setting the red threshold
            red_min = round(gmin * (1 - red_percent_dec))
            IJ.setThreshold(red, red_min, 65535)
            red.updateAndDraw()
            print("Red minimum threshold is {}.".format(red_min))

            # clear ROI Manager
            rm = roi.getInstance()
            rm.reset()
```

```python

            # Set Measurements
            IJ.selectWindow(green.getTitle())
            ms_ops = ms.AREA + ms.MEAN + ms.STD_DEV + ms.MODE + ms.MIN_MAX + ms.LIMIT + ms.MEDIAN + ms.INTEGRATED_DENSITY + ms.LABELS + ms.ADD_TO_OVERLAY
            an.setMeasurements(ms_ops)
            target = wm.getImage(red.getTitle()) # Replace with your target image title
            an.setRedirectImage(target)
            green.updateAndDraw()

            # Analyze Particles
            IJ.selectWindow(green.getTitle())
            an_ops = pa.OVERLAY + pa.ADD_TO_MANAGER + pa.SHOW_RESULTS + pa.INCLUDE_HOLES
            pan = pa(an_ops, ms_ops, rt, min_size, db.POSITIVE_INFINITY, 0, 1)
            pan.analyze(green)
            green.updateAndDraw()
            rt.show("Results")

            # change outline thickness in green channel
            overlay = green.getOverlay()
            if overlay is not None:
                for i in range(overlay.size()):
                    roi_item = overlay.get(i)
                    roi_item.setStrokeWidth(2.5)  # Change thickness here
                    roi_item.setStrokeColor(Color.YELLOW)

            # Save images
            IJ.selectWindow(red.getTitle())
            save_rd = os.path.join(red_folder, "series_{}.jpg".format(x+1))
            IJ.saveAs(red, "JPG", save_rd)

            flattened = green.flatten()
            save_gr = os.path.join(green_folder, "series_{}.jpg".format(x+1))
            IJ.saveAs(flattened, "JPG", save_gr)
            flattened.close()

            # closing green and red channels
            green.close()
            red.close()

            x += 1

        test_chk = GenericDialog("Continue?")
        test_chk.addMessage("Do you want to continue testing?")
        test_chk.enableYesNoCancel("Yes", "No")  # This adds Yes and No buttons
        test_chk.showDialog()

        if test_chk.wasCanceled():
            sys.exit()
```



```python
        elif test_chk.wasOKed():
            tgap = GenericDialog("Testing Green Adjustment Percentage")
            tgap.addMessage("Do you want to change the green threshold adjustment percentage?")
            tgap.addMessage("Current percentage: {}".format(green_percent))
            tgap.enableYesNoCancel("Yes", "No")  # This adds Yes and No buttons
            tgap.showDialog()

            if tgap.wasCanceled():
                sys.exit()

            elif tgap.wasOKed():
                cgap = GenericDialog("Changing Adjustment")
                cgap.addStringField("New green percentage (0 if none)", "")
                cgap.showDialog()

                # establish string variables
                green_percent = cgap.getNextString()

                # establish integer variables
                green_percent = float(green_percent)
                green_percent_dec = green_percent / 100
                continue_testing = True

            else:
                print("User clicked NO")
                continue_testing = True
        else:
            continue_testing = False
            print("Testing Completed")
else:
    print("User clicked NO")

###################### start actual analysis ########################

# Construct subfolder paths properly for screenshots
green_folder = os.path.join(screenshot_gr, "screenshots\\green", file_name_time)
red_folder = os.path.join(screenshot_rd, "screenshots\\red", file_name_time)

# Create folders if they don't exist for screenshots
if not os.path.exists(green_folder):
    os.makedirs(green_folder)
if not os.path.exists(red_folder):
    os.makedirs(red_folder)

# open initial Reuslts Table and ROI
ResultsTable.getResultsTable().reset()
rt = ResultsTable()
roi()
```

```python
x = 0

while x <= end:

    # Set up Bio-Formats import options
    options = ImporterOptions()
    options.setId(lif_path)
    options.setOpenAllSeries(False)  # Load only one series
    options.setSeriesOn(x, True)     # Load only the first series
    options.setSplitChannels(True)      # Split channels into separate windows

    # Load the image
    imps = BF.openImagePlus(options)
    imp = imps[0]  # open the first image of the series (starting point)

    # identifying the channels
    green = None
    blue = None
    red = None

    for imp in imps:
        imp.show()
        title = imp.getTitle()
        print("Opened image:", title)  # debug print to see titles

        if "C=0" in title:
            green = imp
        elif "C=1" in title:
            blue = imp
        elif "C=2" in title:
            red = imp

    # changing color contrasts
    for imp in [green, red, blue]:
        imp.setDisplayRange(0, 3500)
        imp.updateAndDraw()

    # close blue window
    blue.close()

    # Grab the auto minimum
    IJ.selectWindow(green.getTitle())
    ip = green.getProcessor() # access to the pixel data
    ip.setAutoThreshold("Triangle", True, ip.NO_LUT_UPDATE)
    default_gmin = ip.getMinThreshold()
    green_min = default_gmin - round(default_gmin * green_percent_dec)

    green_min = max(50, min(65535, green_min))
```

```python
    green.updateAndDraw()
    gmin = ip.getMinThreshold()

    print("Green minimum threshold is {}.".format(gmin))

    # Setting the red threshold
    red_min = round(gmin * (1 - red_percent_dec))
    IJ.setThreshold(red, red_min, 65535)
    red.updateAndDraw()
    print("Red minimum threshold is {}.".format(red_min))

    # clear ROI Manager
    rm = roi.getInstance()
    rm.reset()

    # Set Measurements
    IJ.selectWindow(green.getTitle())
    ms_ops = ms.AREA + ms.MEAN + ms.STD_DEV + ms.MODE + ms.MIN_MAX + ms.LIMIT + ms.MEDIAN + ms.INTEGRATED_DENSITY + ms.LABELS + ms.ADD_TO_OVERLAY
    an.setMeasurements(ms_ops)
    target = wm.getImage(red.getTitle()) # Replace with your target image title
    an.setRedirectImage(target)
    green.updateAndDraw()

    # Analyze Particles
    IJ.selectWindow(green.getTitle())
    an_ops = pa.DISPLAY_SUMMARY + pa.OVERLAY + pa.ADD_TO_MANAGER + pa.SHOW_RESULTS + pa.INCLUDE_HOLES
    pan = pa(an_ops, ms_ops, rt, min_size, db.POSITIVE_INFINITY, 0, 1)
    pan.analyze(green)
    green.updateAndDraw()
    rt.show("Results")

    sm = ResultsTable.getActiveTable()
    sm.addValue("Series", x + 1)
    sm.addValue("Green", green_min)
    sm.addValue("Red", red_min)
    sm.show("Summary")

    # change outline thickness in green channel
    overlay = green.getOverlay()
    if overlay is not None:
        for i in range(overlay.size()):
            roi_item = overlay.get(i)
            roi_item.setStrokeWidth(2.5)  # Change thickness here
            roi_item.setStrokeColor(Color.YELLOW)

    # Save images
    IJ.selectWindow(red.getTitle())
    save_rd = os.path.join(red_folder, "series_{}.jpg".format(x+1))
    IJ.saveAs(red, "JPG", save_rd)
```

```python

    flattened = green.flatten()
    save_gr = os.path.join(green_folder, "series_{}.jpg".format(x+1))
    IJ.saveAs(flattened, "JPG", save_gr)
    flattened.close()

    # closing green and red channels
    green.close()
    red.close()

    x += 1

# export summary and results
sm = ResultsTable.getActiveTable()
sm.show("Summary")
sm.saveAs(data_sum + "\sum_{}.csv".format(file_name_time))

ResultsTable.getResultsTable()
rt.show("Results")
rt.save(output_dir + "\{}.csv".format(file_name_time))

# End note
por_fin = GenericDialog("End")
por_fin.addMessage("Analysis Completed!")
por_fin.showDialog()
time.sleep(10)
sys.exit()
```

Results

Evaluation of cell area identification

Accurate quantification of specific protein markers in treated cells requires precise identification of individual cells prior to particle analysis [19]. Reliable results depend heavily on proper segmentation, making the careful setting of the minimum fluorescence intensity threshold essential for generating accurate cell masks while minimizing background interference.

To assess the accuracy and consistency of cell area identification, we evaluated the program's performance across multiple immunofluorescence-labeled images. The script's thresholding and particle analysis steps were benchmarked against manually annotated images to ensure reliable segmentation of individual cells (Figure 1A). All images were evaluated by three independent researchers, using both manual and automatic methods. The average individual cell areas obtained by the two methods were compared among the three researchers (Figure 1B). While there was no significant difference among individual observers using either method, there were significant differences between the cell areas measured by manual method than those identified by the automatic analysis (Figure 1B).

To further investigate the source of this variance, we compared the total cell area measurements between the two methods. Although the total cell area measured was slightly higher with the manual method, the difference was not statistically significant (Figure 1C). However, the total cell counts obtained through manual analysis were significantly lower than those detected by the automated approach (Figure 1D). This discrepancy contributed to the larger average cell area observed in manual analyses compared to the automated method (Figure 1E).

These findings indicate that manual analysis tends to overestimate cell areas by merging adjacent cells into larger regions and occasionally incorporating surrounding background pixels, as illustrated in the highlighted region in Figure 1A.

Evaluation of protein intensity quantification

Accurate cell segmentation and thresholding are essential in bioimage analysis, particularly when quantifying fluorescence intensity as a proxy for protein marker expression [1,20]. One key metric used in this process is the mean gray value, which directly reflects the level of protein expression within segmented cell regions. Errors in segmentation, especially those involving overestimation of cell boundaries, can substantially impact this measurement. When masks extend beyond actual cell borders, they may capture background regions with minimal or no signal, thereby lowering the calculated mean gray value for each cell.

To further evaluate the performance of the designed automated mothed in assessing the mean gray values of the protein expression, we compared manual and automated analyses to quantify the protein level of IL36R (Figure 2A). While the average mean-gray value in individual cells did not differ significantly between the two methods (Figure 2B), the total mean gray value across all cells was significantly lower in the manual approach when compared to automated approach (Figure 2C). This discrepancy likely stems from overestimation errors in the manual method. Such overestimations can distort the interpretation of experimental effects and lead to misleading conclusions in downstream analyses.

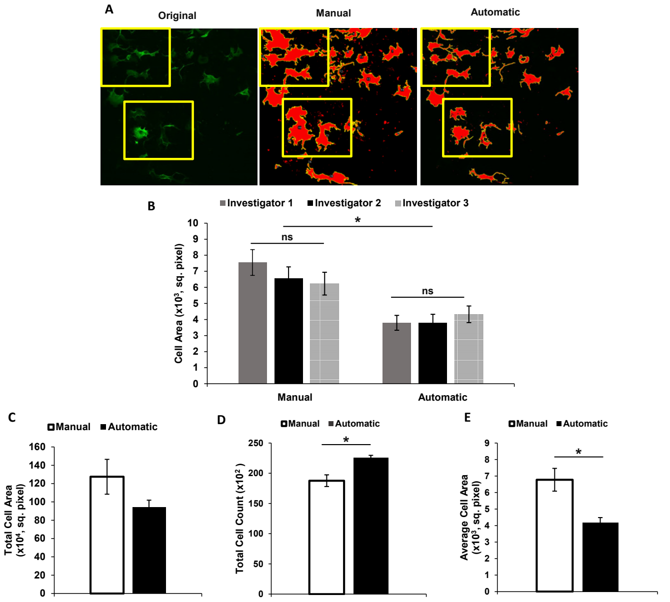


Figure 1: Comparison of VSMC areas identified by independent investigators using manual and automated methods. A. Representative images of VSMCs stained for α-SMA (green) obtained via light microscopy. Seven images were analyzed manually and automatically by three independent investigators. The green signal represents α-SMA fluorescence intensity, while the solid red overlay denotes the threshold-generated mask produced by each method. The green channel emissions were analyzed using both approaches. Yellow boxes highlight examples showing discrepancies in cell boundaries and counts, indicating that manual analysis tended to overestimate the cell boundaries. **B**. A quantitative comparison of individual analyses (n=6) performed using one-way ANOVA with Tukey-Kramer post hoc tests. **C-E.** Method-based differences were further evaluated by grouping manual and automated results (n=3 per method) to compare (**C**) total cell area, (**D**) total cell count, and (**E**) average cell area, using t-tests. The bar graphs display the average ± standard deviation of the analyses. $^*p < 0.05$ or $q > 4.030$.

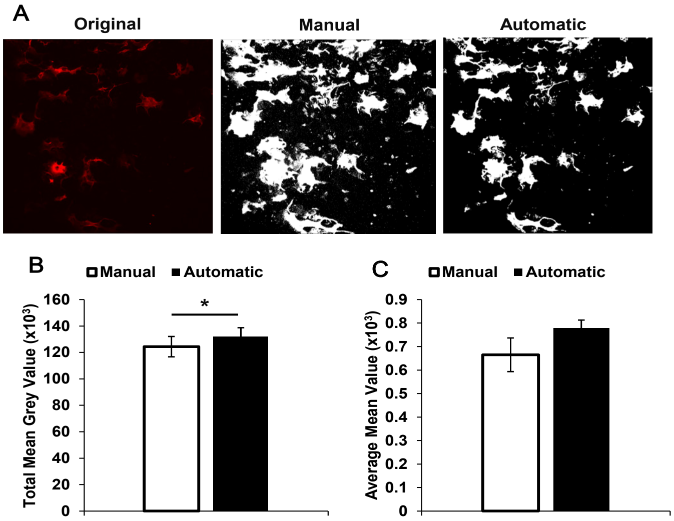


Figure 2: Comparison of quantified protein expression using manual and proposed automatic methods. **A**. Seven images were analyzed manually and automatically by three independent investigators using Fiji. The gradient red signal apparent in the first image represents the original signal intensity of IL36R. In the remaining images, the solid white overlay represents the threshold-generated mask produced by the intensity threshold applied in Fiji. **B-C**. The average and total mean grey values of IL36R were compared between the manual and proposed automated analyses using unpaired t-tests. n=3/group/method. $^*p<0.05$.

Evaluation of bioimage analysis time efficiency

Time consumption remains a practical challenge in analyses of fluorescence-stained images, particularly when dealing with large sample sizes. To evaluate efficiency, we compared the time required by three independent researchers to analyze the same dataset using manual and automated approaches.

As shown in Figure 3, manual analysis of 44 images, comprising about 300-400 cells, took a total of 276 minutes (over 4 hours). In contrast, the automated method completed the same task in just 6 minutes. This result aligns with previous studies that have proposed similar automated approaches to streamline bioimage analysis workflows [21,22].

This demonstrates a substantial improvement in the processing speed of the automated method and underscores its broader implications. Faster analysis not only increases experimental throughput but also enhances analytical rigor. By alleviating time constraints, automated methods allow researchers to process larger datasets and perform additional analyses that may have been impractical with manual approaches.

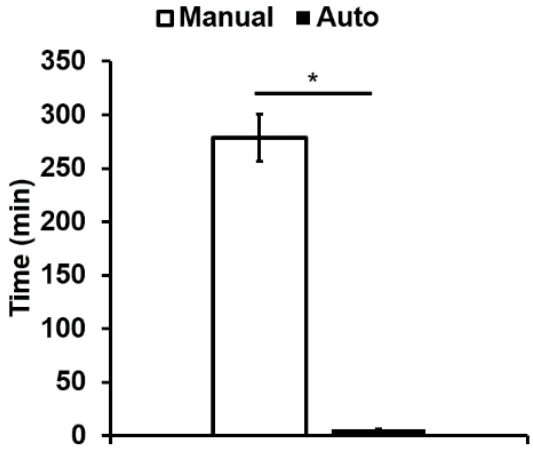


Figure 3: Comparison of time consumption between manual and automated bioimage analysis methods. Forty-four images were analyzed manually and automatically by three independent investigators. n=3/group/methos. The data was compared using an unpaired t-test. *p<0.01.

Discussion

In this study, we developed a semi-automated Jython-based Fiji script to streamline the bioimage analysis of VSMCs following fluorescence immunostaining. This script integrates standard Fiji commands and plugins to automate protein quantification, providing a user-friendly tool that improves efficiency, reduces analysis time, and minimizes inter-user variability.

A source of error in manual image analysis is the overestimation of cell boundaries. To address this, our method employs Fiji's automated Triangle thresholding algorithm, paired with a user-adjustable green intensity scaling parameter. This combination replaces arbitrary thresholding with a data-driven approach, while still allowing user customization to accommodate variability across images. For users preferring a fully automated workflow, the green intensity adjustment can be set to zero, enabling complete reliance on the algorithmic threshold.

Additionally, the script saves processed channel images with overlay masks, giving researchers a visual reference for verifying particle detection and thresholding accuracy. Although this visual verification is inherently subjective, it enhances transparency and supports quality control by allowing users to identify potential segmentation errors or mismatches in thresholding.

Accurate segmentation of the green fluorescence channel is essential not only for quantifying green signal intensity but also for guiding analysis of the red channel [20]. Since red signal quantification depends on identifying VSMCs from the green channel, we implemented a fixed red threshold percentage that remains consistent across samples. To further ensure that relevant red signals are captured, red-channel masks are slightly expanded beyond the boundaries defined by the green-cell segmentation. This strategy enhances consistency, reduces user subjectivity, and improves reproducibility across datasets.

While our approach offers a practical and efficient solution, further validation is necessary due to the inherent variability of bioimage analysis tasks and the ongoing challenge of establishing universal thresholding [20,23]. Validation using synthetic datasets or broader biological conditions would help increase the confidence in the method's robustness and generalizability. Although the script is optimized for our specific experimental setup, its modular workflow can be adapted for other studies involving different cell types, markers, and fluorescence image conditions.

Looking ahead, future development will focus on expanding the script's capabilities. Potential enhancements include batch processing of multiple series simultaneously, integration of customized thresholding methods suited for more complex data, or incorporation of machine learning techniques to enable more adaptive and intelligent segmentation. Such improvements could enhance the script's ability to handle more challenging tasks, such as excluding background noise, correcting for folded or overlapping cells, and refining cell boundaries with greater precision [24].

In summary, our semi-automated Fiji-based script offers a practical, efficient, and reproducible solution for fluorescence image analysis of VSMCs. While initial results are promising, continued refinement and broader validation will be critical for improving threshold accuracy and broadening its applicability across diverse biomedical imaging workflows.

Author declarations

Author contributions

Study conception and design: H.Q., Study conduct and data collection: N.L; K.C., Data analysis: N.L; K.C.. Data interpretation: N.L; K.C.. Drafting of the paper: N.L; K.C., H.Q. Study supervision: H.Q. All authors proofread and approved the final manuscript.

Declaration of competing interest

The authors declare that the research was conducted in the absence of any commercial or financial relationship that could be construed as a potential conflict of interest.

Acknowledgments

The work was supported by US NIH Grant Number HL142291 (H.Q.), HL137962 (H.Q.) and HL115195-01 (H.Q.). We thank Abigail Urbina-Bernal for assisting with partial data analysis.

Data availability

All data needed to evaluate the conclusions are present in the paper, and all raw data and statistical p-values will be provided as requested.

References

1. Cheng J, Wu H, Xie C, He Y, Mou R, et al. Single-cell mapping of large and small arteries during hypertensive aging. J Gerontol A Biol Sci Med Sci. 2024; 79.

2. Olivares-Silva F, Gregorio ND, Espitia-Corredor J, Espinoza C, Vivar R, et al. Resolvin-D1 attenuation of angiotensin II-induced cardiac inflammation in mice is associated with prevention of cardiac remodeling and hypertension. Biochim Biophys Acta Mol Basis Dis. 2021; 1867.

3. She RC, Hammond EH. Utility of immunofluorescence and electron microscopy in endomyocardial biopsies from patients with unexplained heart failure. Cardiovasc Pathol. 2010; 19.

4. Piña R, Santos-Díaz AI, Orta-Salazar E, Aguilar-Vazquez AR, Mantellero CA, et al. Ten approaches that improve immunostaining: a review of the latest advances for the optimization of immunofluorescence. Int J Mol Sci. 2022; 23: 1426.

5. Schneider CA, Rasband WS, Eliceiri KW, et al. NIH Image to ImageJ: 25 years of image analysis. Nat Methods. 2012; 9.

6. Lunde A, Glover JC. A versatile toolbox for semi-automatic cell-by-cell object-based colocalization analysis. Sci Rep. 2020; 10.

7. Robitaille MC, Byers JM, Christodoulides JA, Raphael MP. Automated cell segmentation for reproducibility in bioimage analysis. Synth Biol. 2023; 8.

8. Terheyden JH, Wintergerst MWM, Falahat P, Berger M, Holz FG, et al. Automated thresholding algorithms outperform manual thresholding in macular optical coherence tomography angiography image analysis. PLoS One. 2020; 15.

9. Y H, Z C, B H. Smooth muscle heterogeneity and plasticity in health and aortic aneurysmal disease. Int J Mol Sci. 2023; 24.

10. Persiani E, Ceccherini E, Falleni A, Gisone I, Ippolito C, et al. Ultrastructural and molecular analysis of vascular smooth muscle cells during the switch from a physiological to a pathological phenotype. Biomedicines. 2025; 13: 1127.

11. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, et al. Fiji: an open-source platform for biological image analysis. Nat Methods. 2012; 9.

12. Schwendy M, Unger RE, Bonn M, Parekh SH, et al. Automated cell segmentation in FIJI® using the DRAQ5 nuclear dye. BMC Bioinformatics. 2019; 20.

13. Stansak KL, Baum LD, Ghosh S, Thapa P, Vanga V, et al. PCP auto count: a novel Fiji/ImageJ plug-in for automated quantification of planar cell polarity and cell counting. Front Cell Dev Biol. 2024; 12.

14. Hinkle ER, Essader TO, Gentile GM, Giudice J. ViaFuse: Fiji macros to calculate skeletal muscle cell viability and fusion index. Skelet Muscle. 2021; 11.

15. Peng H, Zhou J, Zhou Z, Bria A, Li Y, et al. Bioimage informatics for big data. Focus on Bio-Image Informatics. 2016.

16. ImageJ Developers. ImageJ API documentation [Internet]. 2025 [cited 2025 Aug 10]. Available from: https://imagej.net/ij/developer/api/help-doc.html

17. SciJava Developers. Bio-Formats JavaDoc API documentation [Internet]. [cited 2025 Aug 10]. Available from: https://javadoc.scijava.org/Bio-Formats/

18. GW Z, WE R, SA L. Automatic measurement of sister chromatid exchange frequency. J Histochem Cytochem. 1977; 25.

19. Wiggins L, O'Toole PJ, Brackenbury WJ, Wilson J. Exploring the impact of variability in cell segmentation and tracking approaches. Microsc Res Tech. 2024; 88.

20. Johnson SJ, Walker FR. Strategies to improve quantitative assessment of immunohistochemical and immunofluorescent labelling. Sci Rep. 2015; 5.

21. Barry DJ, Durkin CH, Abella JV, Way M. Open source software for quantification of cell migration, protrusions, and fluorescence intensities. J Cell Biol. 2015; 209.

22. López AC, Gómez-Pedrero JA, Blanco AMO, Sorzano COS. CellTypeAnalyzer: a flexible Fiji/ImageJ plugin to classify cells according to user-defined criteria. Biol Imaging. 2022; 2.

23. Li X, Zhang Y, Wu J, Dai Q, Li X, et al. Challenges and opportunities in bioimage analysis. Nat Methods. 2023; 20.

24. Jan M, Spangaro A, Lenartowicz M, Usaj MM. From pixels to insights: machine learning and deep learning for bioimage analysis. BioEssays. 2024; 46.