# Assignment 3: Applications of Image Processing to Real-World Problems

Aleksandr Jan Smoliakov

2024–12–20

## 1 Introduction

The report is structured into three sections, each of which describes a different application of image processing to a real-world problem.

In **FISH Signal Counts**, we analyze fluorescence in situ hybridization (FISH) images to detect and quantify genetic mutations in tumor cells. This involves identifying mutations using fluorescent probes and quantifying the signals to understand the mutation distribution within individual cells.

In **Circuit Board Quality Assurance**, we analyze images of circuit boards to detect defects. This involves inspecting soldering regions, traces, and drilled holes using x-ray imaging and image processing techniques.

In **Filled Bottles**, we analyze images of a production line to detect whether bottles are filled to the correct level. This involves detecting the liquid level in bottles using edge detection and intensity analysis.

# Contents

**1 Introduction** **1**

**2 Theoretical Background** **3**
2.1 Thresholding . . . 3
2.1.1 Basic Global Thresholding . . . 3
2.1.2 Optimal Global Thresholding . . . 4
2.1.3 Otsu's Method . . . 4
2.1.4 Some Considerations for Thresholding . . . 4
2.1.5 Improving Global Thresholding . . . 5
2.1.6 Variable Thresholding . . . 5
2.2 Morphological Operations . . . 6
2.2.1 Morphological Erosion . . . 6
2.2.2 Morphological Dilation . . . 6
2.3 Connected Components Labeling . . . 6

**3 FISH Signal Counts** **7**
3.1 Methodology . . . 7
3.1.1 Input Acquisition . . . 7
3.1.2 Thresholding . . . 7
3.1.3 Saving Binary Masks . . . 7
3.1.4 Morphological Operations . . . 7
3.1.5 Connected Components Labeling . . . 7
3.1.6 Border Touching and Cell Filtering . . . 7
3.1.7 Signal Counting . . . 7
3.2 Results . . . 8
3.3 Binary Mask Visualization . . . 8
3.4 Signal Counts and Ratios . . . 9

**4 Circuit Board Quality Assurance** **10**
4.1 Theoretical Background . . . 10
4.2 Methodology . . . 10
4.3 Results . . . 10

**5 Filled Bottles** **11**
5.1 Theoretical Background . . . 11
5.2 Methodology . . . 11
5.3 Results . . . 11

# 2 Theoretical Background

Understanding the real-world applications described in the next sections of this report requires familiarity with several key concepts in digital image processing. This section provides a brief overview of these concepts, namely colour fundamentals, thresholding, morphological operations, image segmentation, and connected components labeling.

## 2.1 Thresholding

Thresholding is a fundamental technique in digital image analysis used to separate objects of interest from the background. The process assigns a binary label to each pixel based on a predefined intensity level called the *threshold*. Mathematically, for a grayscale image $f(x, y)$ and a threshold $T$, the thresholded image $g(x, y)$ can be expressed as:

$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) \geq T, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $g(x, y)$ is referred to as a *binary* image, where 1 typically denotes foreground (e.g. object of interest) and 0 denotes background. Depending on the application, the foreground can represent different elements (e.g. cells in a microscopy image or elements in a circuit board).

This technique can be extended to *multiple thresholding* where more than two classes are defined based on different intensity ranges. Mathematically, for $n$ thresholds $T_1, T_2, \ldots, T_n$, the thresholded image $g(x, y)$ can be expressed as:

$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) \geq T_n, \\ \vdots & \\ n, & \text{if } T_{n-1} \leq f(x,y) < T_n, \\ 0, & \text{otherwise.} \end{cases}$$

However, in the subsequent sections, we focus on the basic binary thresholding technique for simplicity and clarity.

### 2.1.1 Basic Global Thresholding

The simplest form of thresholding assumes a single, constant threshold $T$ that partitions the image into two classes (background and foreground). To determine this threshold, one can examine the intensity histogram of the image. A straightforward approach is to choose $T$ as the *midpoint* between the average foreground and average background intensities, or use domain-specific heuristics.

An improved version of basic global thresholding involves iterative selection:

1. Select an initial estimate for the global threshold $T$.
2. Segment the image into background and foreground using $T$.
3. Compute the mean intensities $\mu_b$ and $\mu_f$ of the background and foreground classes, respectively.
4. Update $T \leftarrow \frac{\mu_b + \mu_f}{2}$.

5. Repeat until $T$ converges (i.e. the change $\Delta T$ is below a predefined threshold).

Although this method is simple, it often performs adequately if the intensity histogram is bimodal.

### 2.1.2 Optimal Global Thresholding

The basic global thresholding methods are intuitive but may not be optimal in all scenarios. They can be sensitive to noise, uneven illumination, or complex object-background distributions.

Optimal global thresholding methods aim to find the threshold value $T$ that best separates the classes in a statistical sense. Typically, this involves maximizing a criterion that quantifies class separability. For instance, one may minimize the overall misclassification error probability by modeling the intensities of the background and foreground with probability density functions (PDFs) and solving for the threshold that minimizes the error.

### 2.1.3 Otsu's Method

Otsu's method is a histogram-based thresholding approach that seeks to maximize the *between-class variance* (or minimize the *within-class variance*) of the thresholded image. Consider an image with an intensity histogram $p(i)$, where $i$ ranges over all possible intensity levels $0 \leq i < L$. For a potential threshold $T$, define:

$$\omega_0(T) = \sum_{i=0}^{T-1} p(i), \quad \omega_1(T) = \sum_{i=T}^{L-1} p(i),$$

$$\mu_0(T) = \sum_{i=0}^{T-1} i\, p(i)/\omega_0(T), \quad \mu_1(T) = \sum_{i=T}^{L-1} i\, p(i)/\omega_1(T).$$

The total mean intensity is:

$$\mu_T = \sum_{i=0}^{L-1} i\, p(i).$$

The between-class variance $\sigma_B^2(T)$ is:

$$\sigma_B^2(T) = \omega_0(T)\left(\mu_0(T) - \mu_T\right)^2 + \omega_1(T)\left(\mu_1(T) - \mu_T\right)^2.$$

Otsu's method exhaustively searches for the threshold $T^*$ that maximizes $\sigma_B^2(T)$. This method also performs well when the histogram exhibits clear foreground-background separation.

### 2.1.4 Some Considerations for Thresholding

Thresholding is a popular image segmentation technique because of its simplicity and effectiveness when the foreground and background intensities are well separated. It serves as a preprocessing step for numerous applications, such as object counting in microscopy images, industrial quality control, and document analysis.

By focusing on separating background and foreground, thresholding simplifies more advanced tasks (e.g. morphological operations) and often leads to computational efficiency.

However, as a simple technique, thresholding is sensitive to noise, illumination variations, and complex object-background distributions. For instance, non-uniform lighting can cause regions of the background to appear brighter or darker, leading to incorrect segmentation using a single global threshold. In practice, controlling or normalizing the illumination is often needed. Some common strategies to address these challenges include adaptive thresholding, smoothing, and edge detection.

### 2.1.5 Improving Global Thresholding

One strategy to improve global thresholding is to *smooth* the image before applying the threshold. Common filters include:

- **Gaussian smoothing**: Convolves the image with a Gaussian kernel to reduce high-frequency noise.
- **Median filtering**: Replaces each pixel with the median of its neighborhood.

By reducing noise, the histogram becomes less distorted, which typically leads to a clearer separation between the foreground and background intensities.

Another approach is to combine thresholding with edge information. Sharp intensity transitions, identified by edge detection filters (e.g. Sobel), mark the boundaries of objects. Integrating these boundaries helps refine the thresholding result:

- **Post-processing**: After global thresholding, apply edge detection and correct misclassified regions near the edges.
- **Pre-processing**: Use an edge map to guide the selection of local or global thresholds (e.g. adjusting $T$ in areas with strong edges).

### 2.1.6 Variable Thresholding

If the lighting or object intensity is not uniform across the entire image, a single global threshold can be inadequate. One solution is to *partition* the image into subregions and apply a (potentially different) global threshold for each subregion. The subregions can be chosen based on:

- **Regular grids**, i.e. dividing the image into blocks of fixed size.
- **Adaptive segmentation** based on image properties (e.g. intensity gradients).

In this strategy, regions with similar illumination conditions can be thresholded more accurately, improving overall segmentation performance.

In *local thresholding*, the threshold at each pixel depends on local image properties, such as the mean or median intensity of its neighborhood. A common approach is to use a weighted sum of the local mean $\mu_{x,y}$ and standard deviation $\sigma_{x,y}$, where for each pixel $(x, y)$:

$$T_{x,y} = a \cdot \sigma_{x,y} + b \cdot \mu_{x,y}.$$

This allows the threshold to dynamically adapt to changes in illumination across the image. Such methods are especially effective in cases where the intensity varies gradually across the scene.

In summary, thresholding is a cornerstone of image segmentation tasks, and the choice between global and local (adaptive) thresholding depends heavily on the application's requirements and the nature of the illumination. Methods like Otsu's enable automatic selection of a global threshold, while local thresholding approaches offer robust solutions under challenging illumination conditions or non-uniform backgrounds.

## 2.2 Morphological Operations

Morphological operations are image processing techniques that probe and transform the structure of *objects* within a binary image. Two primary morphological operations are *erosion* and *dilation*. These operations are often used in combination to refine object boundaries and eliminate noise.

Two additional morphological operations are *opening* and *closing*. Opening is defined as an erosion followed by dilation, while closing is a dilation followed by erosion. These operations are useful for removing small objects and closing gaps in binary images.

### 2.2.1 Morphological Erosion

Erosion removes pixels on object boundaries, effectively shrinking objects in a binary image. It is defined by the minimum value of the neighborhood defined by a structuring element $S$:

$$I \ominus S(x, y) = \min\{I(x + i, y + j) \mid (i, j) \in S\}$$

Erosion is useful for eliminating small artifacts and separating closely connected objects.

Oftentimes, a round structuring element is used to preserve the circular shape of objects during erosion. In this solution, a vectorized implementation of a square structuring element is used for computational efficiency.

### 2.2.2 Morphological Dilation

Dilation adds pixels to the boundaries of objects, expanding them in the image. It is defined by the maximum value of the neighborhood defined by a structuring element $S$:

$$I \oplus S(x, y) = \max\{I(x - i, y - j) \mid (i, j) \in S\}$$

Dilation is employed to close small holes and gaps within objects, ensuring more complete and connected structures. A combination of erosion and dilation operations can be used to refine object boundaries and enhance segmentation results.

Similarly to erosion, a square structuring element is used for computational efficiency in the dilation operation.

## 2.3 Connected Components Labeling

Connected components labeling is a technique used to identify and label distinct objects within a binary image. Each connected group of foreground pixels is assigned a unique label, facilitating individual analysis of objects. This process involves scanning the binary image and grouping pixels based on connectivity criteria (usually 4-connectivity or 8-connectivity).

Mathematically, let $C$ be the set of connected components in a binary image $I$. Each connected component $c_k \in C$ is defined as:

$$c_k = \{(x, y) \mid I(x, y) = 1 \text{ and connected to } c_k\}$$

Once labeled, various properties of each connected component, such as area, perimeter, and intensity, can be computed for further analysis.

# 3 FISH Signal Counts

## 3.1 Methodology

The process for detecting cells and calculating signal counts and ratios involves a series of steps:

### 3.1.1 Input Acquisition

The pipeline accepts three input images corresponding to different fluorescence channels:

- **DAPI Channel**: Stains the cell nuclei, providing a reference for cell localization.
- **Acridine Channel**: Highlights specific cellular components stained with Acridine.
- **FITC Channel**: Represents signals from FITC-labeled molecules.

These images are loaded in TIFF format using the `load_tiff_image` function. The images are then combined into a single RGB image for visualization purposes: DAPI (red), Acridine (green), and FITC (blue).

### 3.1.2 Thresholding

### 3.1.3 Saving Binary Masks

### 3.1.4 Morphological Operations

### 3.1.5 Connected Components Labeling

Post morphological processing, connected components labeling is applied:

- **Labeling Cells**: The cleaned DAPI mask is labeled using `label_connected_components`, identifying individual cells.
- **Labeling Signals**: Similarly, Acridine and FITC masks are labeled.

### 3.1.6 Border Touching and Cell Filtering

### 3.1.7 Signal Counting

The number of Acridine and FITC signals within each cell is counted using the `calculate_signal_count` function. This function iterates over each labeled cell and computes the number of signals present in the corresponding Acridine and FITC masks.

## 3.2 Results

Terminal command to run the analysis:

```
$ $python python/main.py fish-signal-counts \
    input_path_acridine \
    input_path_fitc \
    input_path_dapi
```

Here, the `input_path_acridine`, `input_path_fitc`, and `input_path_dapi` are the file paths to the Acridine, FITC, and DAPI images, respectively. In this analysis, we processed the provided sample images *data/input/BXY-ABCD_Region 002_FOV 00040_Acridine.tif*, *data/input/BXY-ABCD_Region 002_FOV 00040_FITC.tif*, and *data/input/BXY-ABCD_Region 002_FOV 00040_DAPI.tif*.

## 3.3 Binary Mask Visualization

Figure 1 presents the initial binary masks generated from the DAPI, Acridine, and FITC channels. The masks effectively isolate the nuclei and signal regions, providing a clear basis for subsequent analysis.

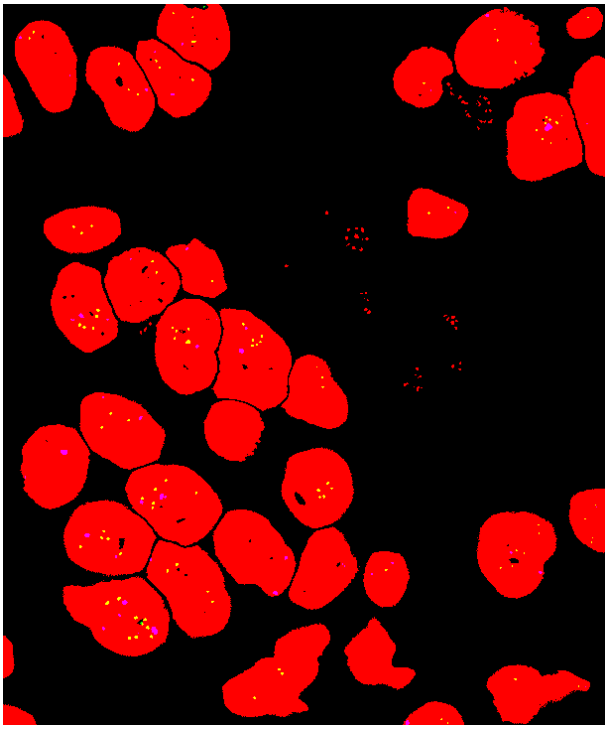

Figure 1: Initial Binary Masks for DAPI (red channel), Acridine (green channel), and FITC (blue channel).

Figure 2 shows the cleaned binary masks after morphological operations and filtering. The processed masks exhibit well-defined cell boundaries and reduced noise, facilitating accurate signal quantification. Cells that touch the image borders are marked with half-intensity.

Figure 2: Cleaned Binary Masks after Morphological Operations.

### 3.4 Signal Counts and Ratios

The pipeline successfully identified and labeled individual cells, enabling the calculation of Acridine and FITC signal counts within each cell. The program identifies 30 complete cells in the provided sample images and computes the signal counts and ratios for each cell.

Table 1 shows the signal counts and ratios for a sample of cells.

Table 1: Signal Counts and Ratios per Cell

| **Cell ID** | **Centroid (X, Y)** | **Area (pixels)** | **Acridine** | **FITC** | **Acridine/FITC** |
|---|---|---|---|---|---|
| 1 | (586.82, 19.07) | 841 | 1 | 0 | - |
| 2 | (497.55, 45.82) | 5015 | 3 | 2 | 1.50 |
| 3 | (44.79, 60.05) | 4161 | 2 | 2 | 1.00 |
| ... | ... | ... | ... | ... | ... |
| 30 | (530.05, 691.11) | 3332 | 2 | 1 | 2.00 |

# 4 Circuit Board Quality Assurance

## 4.1 Theoretical Background

## 4.2 Methodology

## 4.3 Results

# 5 Filled Bottles

## 5.1 Theoretical Background

## 5.2 Methodology

## 5.3 Results