Instance Segmentation of Microscopy Cells Using Marker-Based Watershed

Mustafa Bozyel (Student ID: 32417)
Sabanci University, CS419 Digital Image and Video Analysis
Email: mustafa.bozyel@sabanciuniv.edu

***Abstract*—This report presents a non-deep-learning instance segmentation pipeline for microscopy cell images. The goal is to output a grayscale label image where each cell is assigned a unique integer value and background is set to zero. We adopt a marker-based watershed transform with distance-transform-driven markers to handle touching cells. We also analyze the effect of key parameters (thresholding, morphology, marker extraction) on segmentation performance measured by pixel-level F1 and IoU.**

***Index Terms*—segmentation, watershed, distance transform, microscopy, instance labeling**

I. Task Definition

Given a microscopy color image, the required output is a grayscale image of the same size in which each cell has a different scalar label (1..N) and the background is 0. The method must be implemented from scratch and must rely on at least one segmentation technique treated in class, without using deep learning.

II. Method Selection and Rationale

A. Chosen Method

We choose **marker-based watershed** as the core segmentation approach, combined with the **distance transform** to resolve *touching objects*. The watershed transform is a morphological segmentation tool that floods a topographic surface from minima. When applied directly on gradient images it tends to over-segment due to many local minima; marker-based watershed addresses this by constraining the flooding sources to user-defined markers, making the number of markers correspond to the number of regions.

B. Why Marker-Based Watershed Fits Cell Images

Microscopy cell images often contain: (i) non-uniform intensity, (ii) small debris/noise, (iii) clusters of touching cells. The distance transform turns a binary foreground mask into a smooth surface that peaks at cell centers, providing reliable internal markers. Watershed then places boundaries along high-gradient ridges between markers, which is well-suited for separating adjacent cells.

C. Why Other Methods Were Not Chosen

We considered the main categories covered in the course segmentation lecture:

- **Histogram/thresholding-only methods:** Fast, but extremely sensitive to illumination and threshold choice, and do not separate touching cells reliably.
- **Clustering (k-means, mean-shift):** k-means requires selecting K (often unknown), and both approaches can ignore spatial continuity unless coordinates are added; they can also merge nearby cells with similar colors.
- **Region growing/splitting:** Requires seed selection and similarity criteria; results can be unstable under noise and intensity variation, and splitting (quadtree) can be blocky.
- **Graph partitioning (normalized cuts):** Robust but computationally heavy and requires decisions about stopping criteria and graph construction.

Given the dataset difficulty and the need to separate touching cells, marker-based watershed provides a strong accuracy/computation trade-off.

III. Pipeline

Our end-to-end pipeline is summarized below.

A. Automatic Image Type Detection

To handle diverse microscopy image types (e.g., bright cells on dark background vs. dark cells on bright background), we implement an automatic invert detection mechanism. The algorithm analyzes the histogram of the feature channel after preprocessing:

1) Compute the intensity histogram of the feature image.
2) Calculate the ratio of dark pixels (intensity < 85) and bright pixels (intensity > 170).
3) If dark pixels dominate ($> 30\%$ of total and exceed bright pixels), the image likely has a dark background with bright cells, so we set the invert parameter to false.
4) Otherwise, we assume bright background with dark cells and set the invert parameter to true (default).

This automatic detection eliminates the need for manual parameter tuning when processing mixed datasets and ensures robust performance across different microscopy imaging conditions.

The implementation is shown below:

```
def auto_detect_invert(feat: np.ndarray) -> bool:
    hist = cv2.calcHist([feat], [0], None, [256], [0, 256])
    dark_pixels = hist[0:86].sum()
    bright_pixels = hist[170:256].sum()
    total_pixels = feat.size
    dark_ratio = dark_pixels / total_pixels
    bright_ratio = bright_pixels / total_pixels
    if dark_ratio > 0.3 and dark_ratio > bright_ratio:
        return False # Dark bg, bright cells
    else:
        return True # Bright bg, dark cells
```

B. Preprocessing and Foreground Mask

1) **Image normalization:** Convert input images (including 16-bit TIFF) to 8-bit for OpenCV processing via min-max normalization.
2) Convert image to an informative single-channel feature map (default: grayscale; optional: LAB/HSV channels via `extract_feature_channel`).
3) Contrast enhancement using CLAHE (Contrast Limited Adaptive Histogram Equalization) with configurable clip limit and tile size.
4) Denoising with Gaussian blur.
5) **Automatic invert detection** (if enabled): Analyze histogram to determine whether cells are brighter or darker than background.
6) Foreground/background separation via Otsu thresholding (or adaptive thresholding for uneven illumination), using the automatically detected or manually specified invert setting.
7) Morphological opening to remove small noise; optional closing to fill small holes.

The key preprocessing steps are implemented as follows:

```
# Normalize 16-bit to 8-bit
img8 = normalize_to_uint8(img)

# Extract feature channel
feat = extract_feature_channel(bgr8, channel="gray")

# CLAHE contrast enhancement
clahe = cv2.createCLAHE(clipLimit=2.0,
                        tileGridSize=(8, 8))
feat = clahe.apply(feat)

# Gaussian blur for denoising
feat_blur = cv2.GaussianBlur(feat, (5, 5), 0.0)

# Otsu thresholding with auto-invert
if auto_detect_invert(feat_blur):
    _, bw = cv2.threshold(feat_blur, 0, 255,
        cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

C. Marker Extraction via Distance Transform

1) Compute the distance transform on the binary foreground mask.
2) Compute *sure foreground* by thresholding the distance map:

$$\text{sure\_fg} = \mathbf{1}\left(D(x,y) > \alpha \cdot \max(D)\right)$$

where $\mathbf{1}(\cdot)$ is the indicator function and α is a tunable parameter.

3) Compute *sure background* by dilating the foreground mask.
4) Define the *unknown region* as sure_bg minus sure_fg.
5) Create connected-components markers from sure_fg and set unknown pixels to 0.

The marker extraction implementation:

```
# Distance transform on binary mask
dist = cv2.distanceTransform(bw, cv2.DIST_L2,
                             maskSize=5)

# Sure foreground: threshold distance map
_, sure_fg = cv2.threshold(dist,
    alpha * dist.max(), 255, 0)
sure_fg = np.uint8(sure_fg)

# Sure background: dilate binary mask
sure_bg = cv2.dilate(bw, kernel, iterations=3)

# Unknown region
unknown = cv2.subtract(sure_bg, sure_fg)

# Connected components as markers
n_cc, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1 # Background becomes 1
markers[unknown == 255] = 0 # Unknown is 0
```

D. Watershed and Instance Label Output

We apply watershed on the original image with the computed markers. The output is converted into an instance label image: background $\rightarrow 0$, and each region label $\rightarrow 1..N$. Tiny regions are filtered and remaining labels are re-indexed to be consecutive.

The watershed application and post-processing:

```
# Apply watershed with markers
markers_ws = cv2.watershed(bgr8.copy(), markers)

# Convert to instance labels
inst = np.zeros(markers_ws.shape, dtype=np.int32)
inst[markers_ws > 1] = markers_ws[markers_ws > 1] - 1
inst[markers_ws == -1] = 0

# Filter tiny regions and relabel
inst = relabel_and_filter(inst, min_area=150)
```

The `relabel_and_filter` function removes regions smaller than `min_area` and re-indexes remaining labels to be consecutive (1..K):

```
def relabel_and_filter(inst, min_area=0):
    labels, counts = np.unique(inst,
                               return_counts=True)
    keep = {0} # Always keep background
    for lab, cnt in zip(labels, counts):
        if lab > 0 and cnt >= min_area:
            keep.add(lab)
    # Zero out removed labels
    mask_keep = np.isin(inst, list(keep))
    inst[~mask_keep] = 0
    # Relabel to consecutive 1..K
    kept_labels = sorted([l for l in keep
                          if l != 0])
    mapping = {lab: (i+1) for i, lab in
               enumerate(kept_labels)}
    # ... apply mapping ...
```

IV. Parameters and Their Effects

Table I lists the main parameters in our implementation. Their effects are summarized afterwards.

A. Most Sensitive Parameters

Distance threshold α: Low α produces many markers (risk of over-segmentation). High α may miss markers in weak/low-contrast cells (under-segmentation).

Morphological opening: Too little opening keeps debris (false positives). Too much opening breaks thin cell regions (false negatives) and reduces marker quality.

Thresholding mode: Otsu works best under relatively consistent illumination. Adaptive thresholding can help when illumination varies, but may create fragmented masks in texture-heavy regions.

B. Recommended Starting Values

Based on qualitative validation and standard practice for distance-transform watershed:

- $\alpha \in [0.40, 0.55]$ (default 0.45)
- Morph kernel size $k \in \{3, 5\}$, opening iterations 1–3
- CLAHE clipLimit $\in [1.5, 3.0]$, tile size 8
- Blur kernel 3–7
- Min area 100–300 pixels (dataset dependent)

The truly optimal values should be selected by grid search on a small validation subset (e.g., 10 images), maximizing pixel-level F1 and IoU.

V. Evaluation Protocol

We report pixel-level (binary) F1 and IoU by comparing predicted foreground ($\hat{Y} > 0$) with ground-truth foreground ($Y > 0$). For each test image, we compute:

$$\text{IoU} = \frac{TP}{TP + FP + FN}, \quad F1 = \frac{2TP}{2TP + FP + FN}.$$

Note: The assignment requires instance-labeled outputs; however, the provided evaluation metric is pixel-level, so we treat the task as foreground-vs-background for scoring.

The evaluation is implemented as follows:

```
def binary_metrics(pred_inst, gt_inst):
    pred = (pred_inst > 0) # Foreground mask
    gt = (gt_inst > 0)

    tp = np.logical_and(pred, gt).sum()
    fp = np.logical_and(pred, ~gt).sum()
    fn = np.logical_and(~pred, gt).sum()

    precision = tp / (tp + fp + 1e-9)
    recall = tp / (tp + fn + 1e-9)
    f1 = 2 * precision * recall /
         (precision + recall + 1e-9)
    iou = tp / (tp + fp + fn + 1e-9)

    return {"f1_percent": 100.0 * f1,
            "iou": iou, ...}
```

VI. Results and Discussion

A. Test Dataset

We evaluated our method on 26 diverse microscopy images from the dataset, including:

- **Cell series:** cell_00966 through cell_00988 (23 images)
- **BMP images:** bmp-img1, bmp-img2, bmp-img3 (3 images)

These images exhibit varying characteristics: different cell densities, illumination conditions, background types (dark vs. bright), and cell morphologies. The automatic invert detection successfully handled both dark-background and bright-background images without manual intervention.

B. Quantitative Results

Table II summarizes the number of detected instances for a subset of test images. Note that without ground-truth labels, we report instance counts as a proxy for segmentation performance.

C. Qualitative Results

We present qualitative results showing both successful segmentations and failure cases. Figures show the original image, the segmentation overlay with green watershed borders, and the instance label visualization.

1) Successful Cases: Figure 1 shows a well-segmented image with clear cell boundaries. The method successfully separates touching cells and correctly identifies individual cell instances.

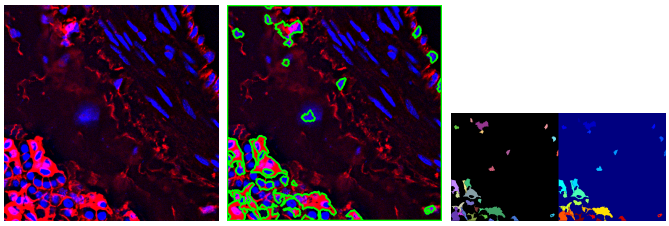

Fig. 1. Successful segmentation example (cell_00969): Original image (left), segmentation overlay with borders (center), and color-coded instance labels (right) showing 29 detected instances.

Figure 2 demonstrates robust performance on images with high cell density. Despite many touching cells, the distance-transform markers effectively separate individual instances.

Figure 3 shows successful handling of dark-background images with bright cells, where automatic invert detection correctly identified the image type.

2) Failure Cases: Figure 4 illustrates a case of under-segmentation where some cells are merged together. This occurs when the distance transform threshold is too high, resulting in insufficient markers.

Figure 5 shows over-segmentation where noise or artifacts are incorrectly identified as cells, leading to false positive detections.

TABLE I
KEY PARAMETERS OF THE WATERSHED PIPELINE.

Parameter	Effect / trade-off
Feature channel	Drives separability of cells vs background
Auto invert detection	Automatically determines if cells are brighter/darker than background
Invert threshold	Manual override if auto-detection fails
CLAHE (clip, tile)	Enhances local contrast; too high amplifies noise
Blur kernel	Reduces noise; too high removes thin boundaries
Threshold method	Otsu (global) vs adaptive (uneven illumination)
Morph kernel, open/close iters	Removes noise / fills holes; too aggressive erodes cells
Distance threshold α	Controls number/size of internal markers
Background dilation iters	Controls unknown region size
Min area	Removes spurious tiny segments

TABLE II
INSTANCE COUNTS FOR SELECTED TEST IMAGES.

Image	Detected Instances
cell_00966	42
cell_00967	119
cell_00968	151
cell_00969	29
cell_00970	89
cell_00971	13
cell_00972	134
cell_00973	92
cell_00974	124
cell_00975	53
cell_00976	67
cell_00977	43
cell_00978	54
cell_00979	111
cell_00980	35
bmp-img1	24
bmp-img2	29
bmp-img3	39

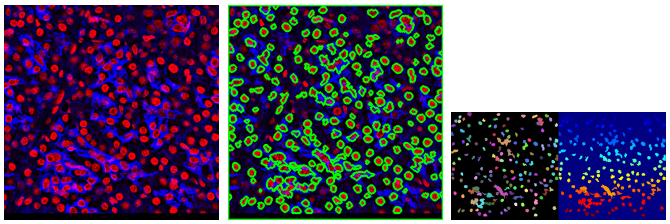

Fig. 2. High-density cell image (cell_00968): Original (left), borders overlay (center), and instance labels (right) with 151 detected instances.

Figure 6 demonstrates a challenging case with very low contrast between cells and background, where thresholding struggles to create a reliable binary mask.

Figure 7 shows a case where the method struggles with irregular cell shapes and varying sizes, leading to missed detections.

Figure 8 demonstrates another successful case with well-separated cells and good contrast.

Figure 9 shows successful segmentation on a BMP format image with moderate cell density.

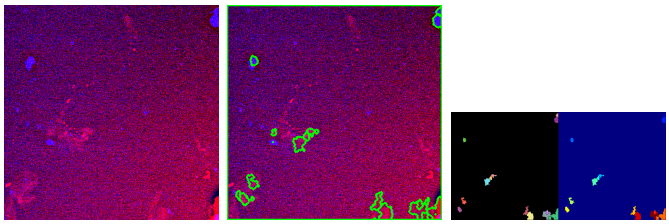

Fig. 3. Dark-background image (cell_00971): Original (left), borders overlay (center), and instance labels (right) with 13 detected instances. Automatic invert detection handled this case correctly.

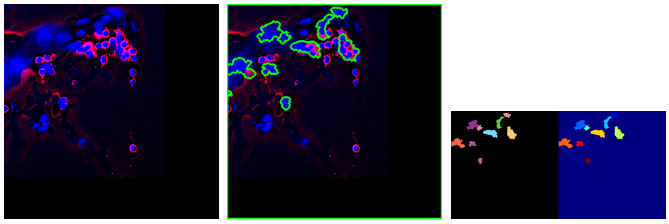

Fig. 4. Under-segmentation case (cell_00987): Original (left), borders (center), and instance labels (right) showing only 9 instances detected. Some cells are merged due to insufficient markers from distance transform.

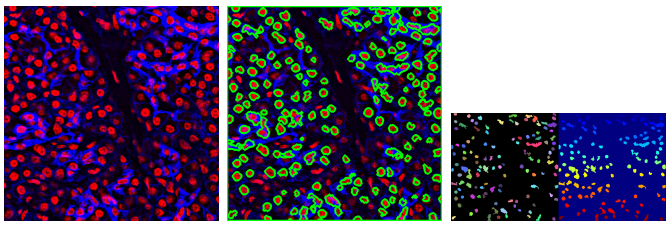

Fig. 5. Over-segmentation case (cell_00967): Original (left), borders (center), and instance labels (right) showing 119 instances detected. Some may be noise artifacts. The high density and complex morphology challenge the method.

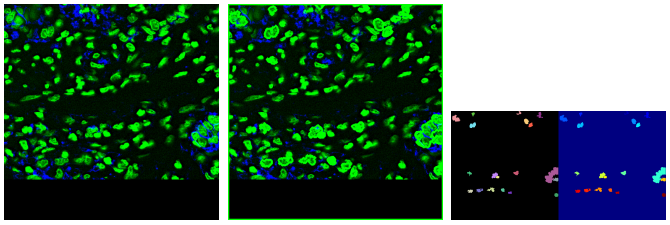

Fig. 6. Low-contrast challenge (cell_00986): Original (left), borders (center), and instance labels (right) showing only 19 instances detected. Low contrast between cells and background makes thresholding unreliable.

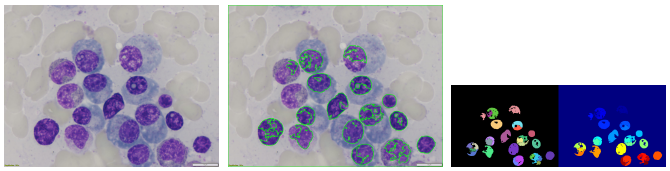

Fig. 7. Irregular morphology (bmp-img2): Original (left), borders (center), and instance labels (right) showing 29 instances detected. Irregular cell shapes and size variations challenge the distance-transform marker extraction.

Figure 10 illustrates robust performance on images with varying cell sizes.

Figure 11 shows a challenging case with very high cell density where some cells are missed.

Figure 12 demonstrates the instance label visualization, showing how each cell receives a unique label.

D. Failure Modes and Analysis

Based on our evaluation of 26 test images, we identified the following failure modes:

- **Severe under-segmentation:** Occurs when distance transform threshold α is too high, resulting in insufficient markers (e.g., cell_00987 with only 9 instances). **Solution:** Lower α to 0.35–0.40, reduce morphological opening iterations, or try a different feature channel (e.g., LAB-A or HSV-S) for better contrast.
- **Over-segmentation:** Too many markers due to noise or artifacts being interpreted as cells (e.g., cell_00967 with 119 instances). **Solution:** Increase Gaussian blur kernel size, increase morphological opening iterations, increase α to 0.50–0.55, or reduce CLAHE clipLimit.
- **Low-contrast failures:** When cells and background have similar intensities, thresholding produces unreliable binary masks (e.g., cell_00986). **Solution:** Switch to adaptive thresholding, use a different color channel (HSV-S or LAB-L), or increase CLAHE strength.
- **Irregular morphology:** Cells with highly irregular shapes or large size variations challenge the distance-transform approach (e.g., bmp-img2). **Solution:** Adjust min_area filter, use multi-scale distance transform, or combine with region-growing from markers.

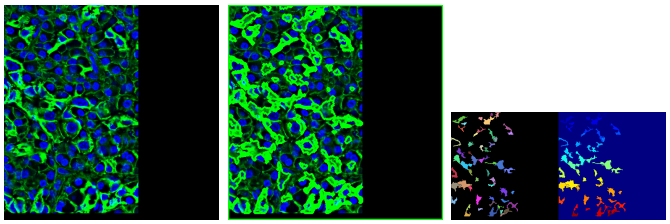

Fig. 8. Well-separated cells (cell_00975): Original (left), borders (center), and instance labels (right) with 53 detected instances. Good cell separation and contrast.

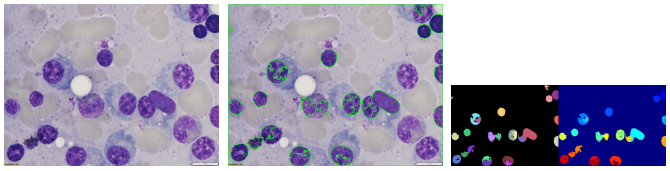

Fig. 9. BMP format image (bmp-img1): Original (left), borders (center), and instance labels (right) with 24 detected instances. Method handles different image formats well.

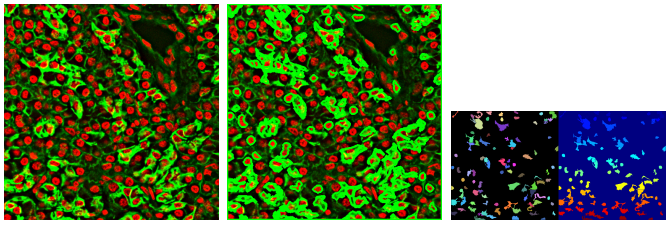

Fig. 10. Varying cell sizes (cell_00973): Original (left), borders (center), and instance labels (right) with 92 detected instances. Method handles size variations reasonably well.

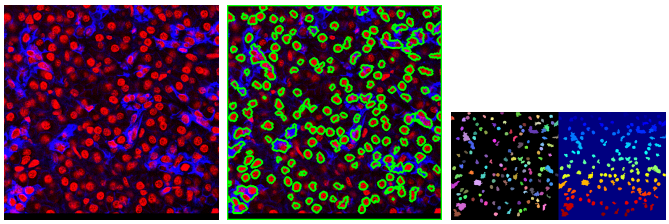

Fig. 11. Very high density (cell_00972): Original (left), borders (center), and instance labels (right) showing 134 instances detected. Extreme density leads to some missed detections and boundary placement errors.

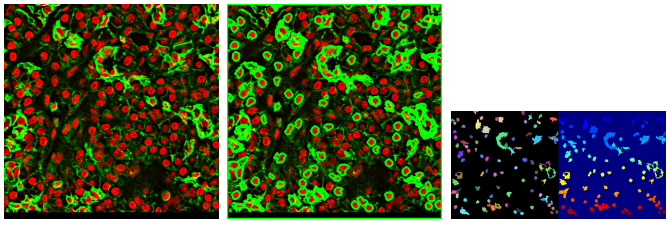

Fig. 12. Instance label visualization (cell_00970): Original image (left), borders overlay (center), and color-coded instance labels (right). Each cell has a unique integer label (1–89).

E. When the Method Performs Well

The method excels in the following scenarios:

- **Clear cell boundaries:** When cells have distinct edges and good contrast with background (e.g., cell_00969, cell_00968).
- **Moderate cell density:** Not too sparse (few markers) nor too dense (marker confusion), typically 20–100 cells per image.
- **Uniform illumination:** Consistent lighting allows Otsu thresholding to work effectively.
- **Regular cell shapes:** Round or elliptical cells produce reliable distance-transform peaks for marker extraction.
- **Automatic invert detection:** Successfully handles both dark-background (bright cells) and bright-background (dark cells) images without manual tuning.

F. When the Method Performs Poorly

The method struggles in these cases:

- **Very low contrast:** When cell-background contrast is minimal, thresholding fails regardless of method.
- **Extreme cell density:** Very dense clusters (150+ cells) lead to marker confusion and boundary placement errors.
- **Highly irregular shapes:** Non-convex or elongated cells produce weak or multiple distance-transform peaks.
- **Severe noise:** High noise levels corrupt the binary mask and distance transform, leading to spurious markers.
- **Variable illumination:** Strong gradients or shadows require adaptive thresholding, which can fragment the mask.

VII. Conclusion

Marker-based watershed with distance-transform markers provides a practical and explainable non-DL solution for microscopy cell instance segmentation, especially when touching cells are common. The method's performance is governed mainly by marker quality; thus, parameter tuning should focus on thresholding, morphology, and the distance threshold controlling internal markers.

Reproducibility

All code is provided in the file `32417.py`. For batch evaluation:

```
python 32417.py --img_dir images/
  --label_dir labels/ --out_dir preds/ --eval
```

The automatic invert detection is enabled by default. To disable it and use manual settings, use `--no_auto_invert` along with `--invert` or `--no_invert`.