Beach Crowd Counter

Technical Summary: Algorithms, Implementation, and Performance

Project Link: https://github.com/guimossibento/project_1_image_and_video_analysis

Emanuel Hegedus and Guilherme Mossi Bento

1. Introduction

This project implements a **people counter for beach images** using only **classical computer vision** in a Jupyter notebook. The goal is to estimate how many people appear in fixed camera images of a beach.

The system does **not** use deep learning. Instead, it uses an effective pipeline:

- Contrast enhancement with CLAHE and adaptive gamma correction
- Polygon-based region masking to isolate the beach area
- HSV color filtering to remove sand regions
- Morphology and adaptive thresholding to obtain a binary mask
- Blob detection with geometric filters to approximate each person
- Global metrics (MAE and error percentage) to evaluate performance

2. Pipeline Overview

The detection pipeline processes each image through the following stages:

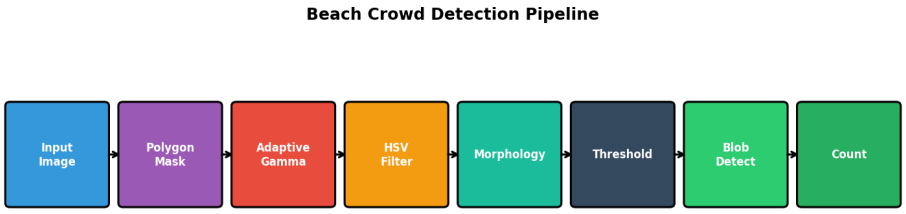


Figure 1: Processing pipeline from input to detection

The method works image by image. For each beach image, the notebook applies the same sequence of operations:

1. **Load image and ground truth** (manual head annotations).
2. **Apply polygon mask** to isolate the region of interest.
3. **Preprocess image**: CLAHE, adaptive gamma, Gaussian blur.
4. **Remove sand** using an HSV-based color mask.
5. **Create binary mask** with morphology and adaptive thresholding.
6. **Detect blobs** with SimpleBlobDetector and count them.

Parameters are first explored with an interactive widget, then a batch loop applies fixed parameters to all images.

3. Algorithms and Implementation

3.1 Main Detection Algorithm

```
ALGORITHM: BeachCrowdDetection(image, polygon, params)
INPUT:  image (RGB), polygon (ROI vertices), params (thresholds)
OUTPUT: count (detected people)

1.  masked ← ApplyPolygonMask(image, polygon)
2.  enhanced ← CLAHE(masked, clip_limit)
3.  γ ← ComputeAdaptiveGamma(enhanced, target_brightness)
4.  normalized ← ApplyGamma(enhanced, γ)
```

```
5.  blurred ← GaussianBlur(normalized, kernel_size)
6.  hsv ← ConvertToHSV(blurred)
7.  sand_mask ← (hsv.S < s_max) AND (hsv.V > v_min)
8.  non_sand ← blurred WHERE NOT sand_mask
9.  gray ← ConvertToGray(non_sand)
10. morphed ← Close(Open(gray, kernel), kernel)
11. binary ← AdaptiveThreshold(morphed, block_size, C)
12. blobs ← BlobDetector(binary, area, shape filters)
13. RETURN length(blobs)
```

3.2 Adaptive Gamma Correction

```
FUNCTION ComputeAdaptiveGamma(image, target):
    current ← median(grayscale(image))
    γ ← log(target/255) / log(current/255)
    RETURN clamp(γ, 0.2, 3.0)
```

3.1 Image Loading and Ground Truth

Images are read with OpenCV and converted from BGR to RGB. Ground-truth annotations are stored in a CSV file with one point per person (x, y coordinate of the head). The ground-truth count for each image is the number of rows in the CSV corresponding to that file.

3.2 CLAHE (Contrast Enhancement)

CLAHE (Contrast Limited Adaptive Histogram Equalization) is applied to enhance local contrast. Beach images often have bright sand and dark shadows. CLAHE increases contrast in dark areas where people may be, without producing excessive noise. The *clahe_clip* parameter controls the contrast limiting.

3.3 Polygon Mask (Region of Interest)

Instead of a simple top mask that removes a fixed percentage of the image, the system now uses a **polygon-based mask** loaded from a JSON file. This polygon precisely defines the beach area where people can appear.

Advantages over top mask: The polygon can follow irregular boundaries (buildings, rocks, water line) that a horizontal cut cannot. It excludes areas like the sea, sky, and structures more accurately, reducing false positives from waves, clouds, or distant objects. The polygon is defined once using annotation tools and applied consistently to all images from the same camera.

3.4 Adaptive Gamma Correction

A key improvement in this version is **adaptive gamma correction**. Instead of using a fixed gamma value, the system automatically calculates the optimal gamma based on each image's current brightness:

$$\gamma = log(target / 255) / log(current / 255)$$

Where *current* is the median brightness of the image and *target* is the desired brightness (default: 160). This approach normalizes brightness *before* applying HSV filters, so the same color thresholds work consistently across varying lighting conditions (sunny, overcast, morning, evening).

Benefits: No manual gamma tuning per image; HSV sand filters work consistently; more robust detection in mixed lighting datasets.

3.5 Gaussian Blur

A Gaussian blur with a small kernel (e.g., 3×3 or 5×5) reduces high-frequency noise from sand texture while preserving the general shape of people, making thresholding and blob detection more stable.

3.6 HSV Sand / Non-Sand Mask

The image is converted to HSV color space. Sand is identified by low saturation (*hsv_s_max*) and high brightness (*hsv_v_min*). Pixels meeting both criteria are classified as sand and removed. People, umbrellas, and towels typically have higher saturation or different brightness than sand.

3.7 Morphology and Adaptive Thresholding

The non-sand image is converted to grayscale. Morphological **opening** removes small isolated noise, while **closing** fills gaps inside blobs. **Adaptive Gaussian thresholding** computes a local threshold for each pixel, handling non-uniform beach lighting better than global thresholding.

3.8 Blob Detection with SimpleBlobDetector

OpenCV's SimpleBlobDetector identifies candidate blobs using four filters:

- **Area** (min_area, max_area): Removes very small blobs (noise) and very large blobs (umbrellas, merged groups).
- **Circularity**: Keeps roughly circular/elliptical blobs (heads, upper bodies).
- **Convexity**: Prefers compact shapes, rejects very concave ones.
- **Inertia ratio**: Controls elongation, balancing standing vs. sitting people.

4. Interactive Tuning and Batch Evaluation

The notebook uses *ipywidgets* as a parameter search tool. The user selects sample images and adjusts sliders for target_brightness, clahe_clip, hsv_s_max, hsv_v_min, morph_size, and blob filter parameters. Each change runs the full pipeline and shows detections with per-image error.

Once good parameters are found, a **batch loop** processes all images with fixed parameters to compute global performance metrics.

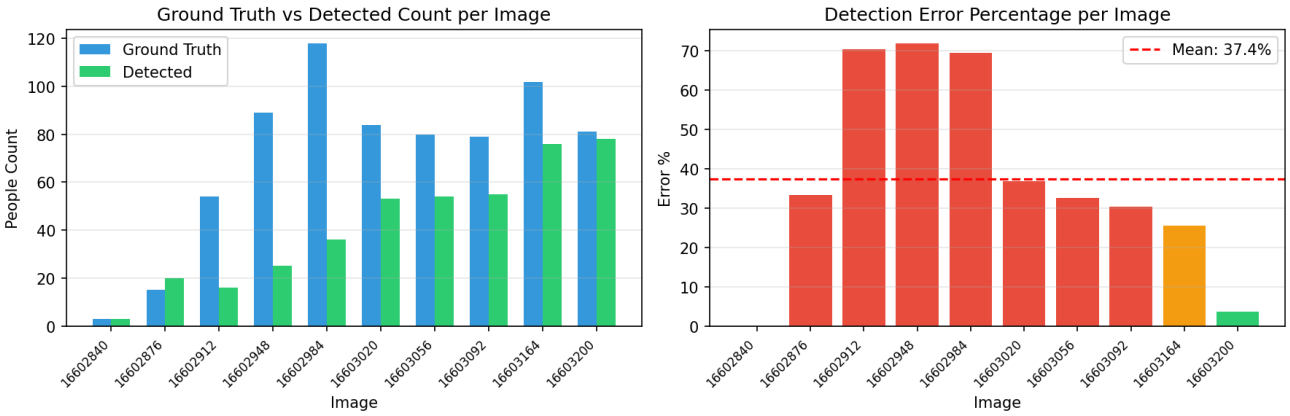
5. Performance Metrics

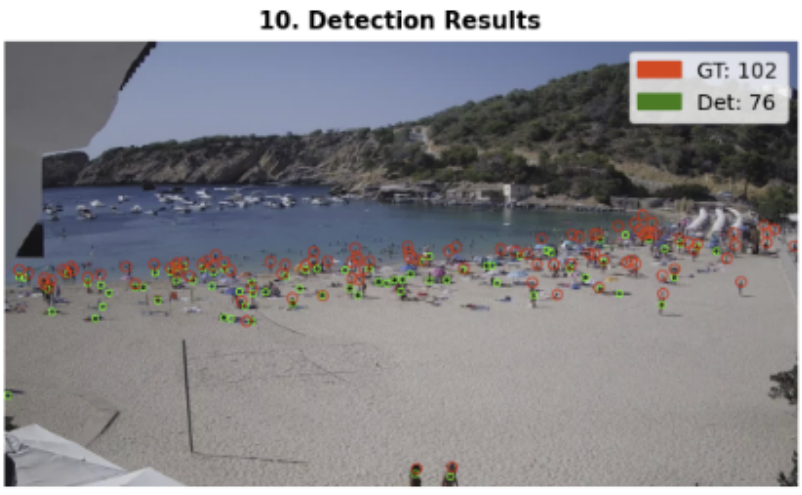
Evaluation is done at the image level. For each image: ground-truth count, detected count, absolute error, and error percentage. The main metric is

Mean Absolute Error (MAE) $= (1/N)\ \Sigma\ |C\_gt - C\_det|$

We also report mean error percentage, describing how large the error is relative to the real number of people.

5.1 Performance Result and Example of Detection





```
======================================================================
QUANTITATIVE RESULTS SUMMARY
======================================================================
Image                GT    Det    MAE   Error%
----------------------------------------------------------------------
1660284000            3      3      0     0.0%
1660287600           15     20      5    33.3%
1660291200           54     16     38    70.4%
1660294800           89     25     64    71.9%
1660298400          118     36     82    69.5%
1660302000           84     53     31    36.9%
1660305600           80     54     26    32.5%
1660309200           79     55     24    30.4%
1660316400          102     76     26    25.5%
1660320000           81     78      3     3.7%
----------------------------------------------------------------------
MEAN               70.5   41.6   29.9    37.4%
======================================================================

GLOBAL METRICS:
  Total Ground Truth: 705 people
  Total Detected:     416 people
  Mean Absolute Error (MAE): 29.90 people/image
  Mean Squared Error (MSE):  1518.70
  Mean Error Percentage:     37.4%
```

6. Design Choices, Limitations, and Conclusion

Key improvements in this version:

- **Adaptive gamma** eliminates manual brightness tuning and handles varying lighting conditions automatically.
- **Polygon masking** provides precise region-of-interest definition, superior to simple top cropping.

Remaining limitations:

- **Occlusion:** In crowded regions, people can merge into single blobs, causing under-counting.
- **Similar shapes:** Some objects (umbrellas, signs) may produce people-like blobs.
- **Extreme lighting:** Very strong shadows or reflections can still challenge the pipeline.

Despite these issues, the approach works well for **sparse to medium beach crowds**, providing reasonable people counts with low computational cost and full transparency. The combination of adaptive gamma and polygon masking significantly improves robustness.