Author: Josep Riballo Moreno
Universitat Autónoma de Barcelona UAB
July 25, 2025

Riballo Symmetry Axis Labeling Application

Abstract

This report details the development and functionalities of the "Riballo Symmetry Axis Labeling Application," a Python-based desktop tool designed to facilitate the human annotation and perceptual validation of symmetry axes in images. Developed using Tkinter for the graphical user interface, alongside Pillow, NumPy, and SciPy for image and data processing, the application addresses the challenge of subjective ground truth generation in symmetry detection research. It provides flexible data loading options, a streamlined labeling workflow, and a robust mechanism for calculating inter-rater agreement metrics (Positive Agreement - PA, and Negative Agreement - NA) from multiple user sessions, thereby enabling a more rigorous perceptual evaluation of symmetry detection algorithms.

1. Introduction

Symmetry is a ubiquitous feature in both natural and artificial environments, playing a fundamental role in fields such as computer vision, pattern recognition, and computer graphics. The accurate detection of symmetry axes in images is a challenging problem, particularly in unconstrained real-world scenarios where variations in appearance, noise, and occlusions are common. While computational models for symmetry detection have advanced significantly, their ultimate validation often relies on comparison with human perception, which can be inherently subjective.

The associated research, particularly the methodology described in Section D of the paper "Combining Appearance and Gradient Information for Image Symmetry Detection" by Gnutti, Guerrini, and Leonardi [1], highlights the importance of perceptual user tests to assess algorithm performance. These tests provide a complementary perspective to traditional precision-recall metrics, directly evaluating how well an algorithm's output aligns with human visual perception of symmetry.

This project aims to provide a dedicated desktop application to streamline such perceptual user tests. The application is designed to:

- Efficiently present images with detected symmetry axes to human annotators.
- Collect structured responses regarding the acceptability and principal nature of these axes.
- Manage various input data formats for axes (MATLAB files, CSV files).
- Automate the calculation of inter-rater agreement metrics (PA and NA) across multiple annotators, thereby providing a quantitative measure of consistency in human judgments, which is crucial for validating subjective ground truth.

2. System Architecture

The application is structured into several modular Python scripts, promoting maintainability and clarity of functionality:

- **main.py:** The primary entry point of the application. It initializes the Tkinter root window and instantiates the AxisLabelingApp class.
- **gui.py:** Contains the core logic for the Graphical User Interface. It manages all visual elements, user interactions (button clicks, radio button selections), image display, and orchestrates the workflow of labeling sessions and score calculation.
- **data_manager.py:** Handles all data-related operations. This includes loading axis data from various file formats (.mat, .csv), implementing the axis selection logic for labeling sessions, recording user responses,

saving session results, and performing the PA/NA agreement calculations.
- **config.py:** Stores all configurable parameters and constants, such as image counts, session question limits, classification thresholds, and directory paths. This centralizes settings for easy modification.

The application follows a clear data flow: input data (images and axis information) is loaded by data_manager.py, presented and collected via gui.py, and results are saved back by data_manager.py. PA/NA calculations then process these saved results.

3. Core Functionalities

3.1 Data Management and Loading

The data_manager.py module is responsible for robustly loading axis data, supporting multiple formats and organizational structures:

- **Flexible Input Formats:**
  - **MATLAB Files** (.mat): The application can load axis data from individual .mat files, typically one per image. It expects a specific naming convention (e.g., "…"refs_001.mat) and a variable named img_detected_refs within each .mat file.
  - **CSV Files** (.csv): Offers two modes for CSV input:
    - **Single CSV File:** A single CSV file containing all axis data for all images.
    - **Multiple CSV Files:** Individual CSV files, one per image (e.g., refs_001.csv), each containing axis data specific to that image.
- **User Selection:** At application startup, the user is prompted to choose the desired data format (.mat or .csv) and, if CSV is selected, whether it's a single or multiple file setup.
- **Data Structure Consistency:** The loader verifies the presence of required columns/variables and attempts basic type conversions (e.g., string to float) to ensure data integrity. Warnings are issued for missing files or malformed data.

3.2 Image and Axis Rendering

The gui.py module handles the visual presentation of images and their corresponding symmetry axes:

- **Two-Stage Image Scaling:**
  1. **Pre-scaling:** This preliminary scaling step adjusts the image's dimensions based on its smallest side (width or height) relative to a MIN_DIM_TARGET (e.g., 200 pixels) defined in config.py. This is critical if the axes were measured on a version of the image that was either smaller or larger than the original image being displayed.

     If PRE_SCALING_GREATER is True in config.py: This scenario applies when the axes were measured on a smaller, resized version of the original image. If the smallest dimension of the original image is larger than MIN_DIM_TARGET, the image is scaled down so that its smallest dimension becomes exactly MIN_DIM_TARGET. This prevents very large images from being processed inefficiently and standardizes the input size for the next stage.

     If PRE_SCALING_SMALLER is True in config.py: This scenario applies when the axes were measured on a larger, resized version of the original image. If the smallest dimension of the original image is smaller than MIN_DIM_TARGET, the image is scaled up so that its smallest dimension becomes exactly MIN_DIM_TARGET. This prevents very small images from being displayed as tiny, unviewable

elements.

This pre-scaling maintains the image's aspect ratio.

2. **Final Scaling:** After pre-scaling, the image is further scaled to fit optimally within the Tkinter canvas dimensions, ensuring it is fully visible and centered while preserving its aspect ratio.

- **Precise Axis Drawing:** Axis coordinates (x1, y1, x2, y2) retrieved from the input data are precisely scaled using the final scaling. This ensures the drawn axis accurately overlays the corresponding feature on the displayed image.

3.3 Axis Selection and Classification

The data_manager.py module implements the logic for selecting axes for a labeling session and classifying them based on their properties:

- **Session Axis Selection:** For each labeling session, a configurable number of axes (QUESTIONS_PER_SESSION from config.py) is selected.
- **Balanced Selection:** The selection process aims to achieve a balanced distribution of axis types (YY, YN, NN) based on TARGET_YY_PERCENTAGE and TARGET_YN_PERCENTAGE defined in config.py.
- **Axis Classification Rules:** Axis are categorized based on their axis_row_index (their order in the input matrix) and their confidence score (the 5th column in the input data):
  - **YY (Yes/Yes):** Always the first detected axis (axis_row_index = 0), which is assumed to have a confidence score of 1.0. This typically represents the most prominent or "principal" symmetry axis.
  - **YN (Yes/No):** Exclusively the second detected axis (axis_row_index = 1) if its confidence score is greater than THRESHOLD_NEAR_ONE (e.g., 0.8) and less than 1.0. This implies it's a good symmetry but not the primary one.
  - **NN (No/No):** Any other axis (i.e., any axis with axis_row_index > 1, or the second axis if it doesn't qualify as YN) is classified as NN. This rule ensures that all axes are categorized as YY, YN, or NN, eliminating the ambiguous 'UNKNOWN' type from the results.

3.4 Labeling Interface and Workflow

The gui.py module manages the user's interaction during a labeling session:

- **Question Presentation:** Displays two clear questions (Q1: acceptable symmetry? Q2: principal symmetry?) with "Yes"/"No" radio button options.
- **Automated Progression:**
  - If Q1 is answered "No", Q2 is automatically set to "No", and the application automatically advances to the next axis.
  - If Q1 is answered "Yes", Q2 is enabled for user input. Answering Q2 (either "Yes" or "No") automatically triggers advancement to the next axis.
- **Navigation:** "Next Axis" and "Previous Axis" buttons allow forward and backward movement through the session.
- **Session Management:** The application tracks progress (Progress: X/Y) and provides a "Finish Session" button to end the session prematurely.

3.5 Session Results Saving

User responses from each labeling session are meticulously recorded:

- **Output Format:** Results are saved as separate CSV files in the results/ directory.
- **Filename Convention:** Each file is named using a timestamp and a configuration string (e.g., session_results_YYYYMMDD_HHMMSS_Q50_TYY40_TYN30_TN80_TF30.csv). This configuration string embeds key session parameters (QUESTIONS_PER_SESSION, TARGET_YY_PERCENTAGE, TARGET_YN_PERCENTAGE, THRESHOLD_NEAR_ONE, THRESHOLD_FAR_FROM_ONE), which is vital for later consistency checks during PA/NA calculation.
- **Columns: Each row in the results CSV includes:** timestamp (of the response), image_base_name, axis_row_index, score (the original confidence score, formatted to 1 decimal place), q1_answer, q2_answer, and expected_type (the pre-classified category of the axis).
- **Overwriting on Back-Navigation:** If a user navigates back to a previously answered axis, changes their response, and then advances, the new response for that specific axis (identified by image_base_name and axis_row_index) will overwrite the old one in the session's internal results list.

4. Perceptual User Test Analysis (PA/NA Calculation)

This functionality is designed to quantitatively assess the consistency of human judgments across multiple labeling sessions, addressing the inherent subjectivity in human-generated ground truth for symmetry detection.

4.1 Motivation

As highlighted in the associated research paper [1], evaluating symmetry detection algorithms solely based on traditional precision-recall curves on limited datasets can be insufficient due to the subjective nature of human perception of symmetry. Perceptual user tests, where human annotators validate algorithm outputs, provide a crucial complementary perspective. To quantify the agreement among multiple human annotators, inter-rater agreement metrics like Positive Agreement (PA) and Negative Agreement (NA) are employed.

4.2 Calculation Method

The PA/NA calculation follows the methodology described in Section D of the paper [1], adapted for multiple annotators:

- **Input:** The calculation uses multiple CSV result files from the results/ directory. Each CSV file is assumed to represent the labeling session of a single human annotator.
- **Configuration Consistency Check:** Before performing any calculation, the system rigorously checks if all selected result CSVs were generated under the exact same configuration (by comparing the CONFIG string embedded in their filenames). If any inconsistency is detected, the calculation is aborted, and an error message is displayed, emphasizing the importance of comparing results from comparable experimental setups.
- **Pooling Results:** Responses from all selected annotators are pooled and grouped by unique axis (identified by image_base_name and axis_row_index).
- **Pairwise Agreement Tally:** For each unique axis, all possible pairs of annotators are considered. For each pair, their responses to Q1 and Q2 are categorized into:
  - **YY (Yes/Yes):** Both annotators said "Yes".
  - **YN (Yes/No):** First annotator said "Yes", second said "No".
  - **NY (No/Yes):** First annotator said "No", second said "Yes".
  - **NN (No/No):** Both annotators said "No".

  These counts (total_yy_q1, total_yn_q1, etc.) are accumulated across all axes and all annotator pairs.
- **PA and NA Formulas:**
  - **Positive Agreement (PA):** $PA = 2 \cdot YY / (2 \cdot YY + YN + NY)$
    - Measures the proportion of times annotators agree on a "Yes" response, relative to all pairs where at least one "Yes" was given.

- **Negative Agreement (NA):** $NA = 2 \cdot NN / (2 \cdot NN + YN + NY)$
  - Measures the proportion of times annotators agree on a "No" response, relative to all pairs where at least one "No" was given.
- These formulas are applied separately for Q1 and Q2. The "final score" displayed in the application is the average of PA_Q1 and PA_Q2, and NA_Q1 and NA_Q2 respectively.

A score closer to 1 indicates stronger agreement.

- **Robustness:** The calculation handles cases where certain questions might not have been answered by all annotators or where denominators might be zero (to avoid division by zero errors).

4.3 Output

The results of the PA/NA calculation are saved and displayed:

- **Display:** A message box shows the calculated PA and NA scores, along with the number of result files (annotators) used.
- **Output File:** The scores are appended as a new row to a CSV file named score.csv.
- **Location:** /score/score.csv
- **Columns:** timestamp (of the calculation), num_results_used, pa_score (formatted to 4 decimal places), na_score (formatted to 4 decimal places).

4.4 User Interaction

The PA/NA calculation functionality is integrated into the user workflow:

- **Startup Button:** A "Calculate Score from Existing Results" button is available on the initial welcome screen, allowing users to perform calculations without starting a new labeling session.
- **Post-Session Prompt:** After a labeling session is completed (either by finishing all axes or clicking "Finish Session"), the user is prompted with a question asking if they wish to calculate the PA/NA score immediately.

5. Conclusion

The "Riballo Symmetry Axis Labeling Application" provides a comprehensive tool for conducting perceptual user tests in the domain of symmetry detection. By offering flexible data loading, a streamlined labeling process, and robust inter-rater agreement calculation (PA/NA), it directly supports the validation of computational symmetry algorithms against human perception. This approach addresses the inherent subjectivity in symmetry ground truth generation, enabling researchers to gain deeper insights into the alignment between algorithmic performance and human visual interpretation. The modular design ensures maintainability, while the detailed logging and output formats facilitate further analysis of both individual labeling sessions and aggregated perceptual scores.

6. References

[1] A. Gnutti, F. Guerrini, and R. Leonardi, "Combining appearance and gradient information for image symmetry detection," *IEEE Transactions on Image Processing*, vol. 30, pp. 3940–3952, 2021, doi: 10.1109/TIP.2021.3079391.