Automatic Crokinole Round Scoring from a Single Photo Project Report

William Bushell Rudra Patel

December 5, 2025

Abstract

An interpretable, classical computer vision pipeline was implemented to compute Crokinole round scores from a single photograph. The pipeline locates and validates a Crokinole board, detects discs, assigns per disc scores (0, 5, 10, 15, 20) with handling of discs touching ring boundaries, clusters discs by colour into two teams, and produces per image overlay visualizations and a results CSV for audit. This report documents algorithms, configuration and design choices, evaluation methodology, results on the provided image set, observed failure modes, and prioritized recommendations.

1 Introduction and objectives

Given a single photograph of a finished Crokinole round, the system must determine each player's final score according to official rules. Requirements:

1. Validate that the image contains a Crokinole board and reject non board images.
2. Detect visible discs and determine each disc's scoring region (0, 5, 10, 15, 20).
3. Apply the rule that a disc touching a ring line receives the lower adjacent score.
4. Partition discs into two player groups (colour) and compute per player totals, accept manual counts of removed 20 point discs.
5. Produce overlay visualizations and a per image results CSV for audit and evaluation.

Approach: a geometry and photometry driven pipeline (edge detection, Hough circle analysis, perimeter sampling, Lab clustering). The implementation is in `Crokinole.py` and is driven by Jupyter notebooks that produce overlays and a results CSV.

2 Data and experimental setup

The image set used for development and evaluation is stored in `images/` in the repository. Notebooks process the images and write overlays and a per image summary CSV to `results/testset_results.csv`. Ground truth available in the repository is team level totals per image, per disc ground truth annotations are not available nor implemented.

Reproducibility: environment is specified by `requirements.txt`. The notebook `testset_script.ipynb` calls the pipeline entrypoint `run_complete_pipeline(img_path, config, team1_20s, team2_20s)` for each image and writes outputs.

3 Pipeline overview

Stages (implemented in `Crokinole.py`):

1. Preprocessing (resizing, RGB conversion).
2. Edge detection (Canny).
3. Ring detection and board validation (adaptive Hough + ratio based verification).
4. Scoring region mask creation (per pixel values 0,5,10,15,20).
5. Disc detection (brightness adaptive Hough).
6. Colour clustering for team assignment (Lab mean patch, K=2).
7. Per disc scoring by perimeter sampling and application of the lower of touched values rule.
8. Aggregation of visible scores and addition of manual 20 counts to compute final totals.

Configuration values are centralized in `config.py`. The most significant parameters used by the pipeline are given in the algorithm subsections below.

4 Ring detection: `detect_board_and_rings`

Inputs and outputs

- **Input**: binary edge map (Canny) of the preprocessed image.
- **Output**: `board_result` containing per ring centres/radii and a refined board centre, or `None` if verification fails.

Algorithm (summary)

1. Compute Canny edges with thresholds:

```
canny_low_threshold = 0.1
canny_high_threshold = 0.2
edge_sigma = 2
```

2. Build an adaptively sampled radii vector so that approximately two hundred radii cover the plausible ring size range for the image.
3. Run the circular Hough transform and extract peaks using `hough_circle_peaks`.
4. Consider the largest peaks as candidate outer rings (`ring_5`). For each candidate:
   - Compute expected inner ring radii from configured ratios, e.g. `ring_ratios = {ring_5:1.00, ring_10:0.66, ring_15:0.33, center:0.05}`
   - For each expected inner radius, search among detected circles for a match within tolerances:

```
radius_tol = max(5 px, 0.15 * expected_r)
center_tol = max(5 px, 0.03 * board_radius)
```

5. If a coherent set of rings is found, compute a refined centre (average of ring centres) and return `board_result`.

Design decisions and rationale

- Adaptive radii sampling reduces Hough runtime while maintaining scale coverage.
- Ratio based verification enforces global geometric consistency and reduces false positives from incidental circles.
- Tolerances combine absolute floors (5 px) with relative fractions (15% for radii, 3% for centres) to handle both large and small images.
- The function rejects images that fail verification to avoid unreliable downstream scoring.

Complexity and failure modes

- Hough complexity scales with image area times number of radii, adaptive sampling mitigates cost.

- Failure modes: weak edges (glare/underexposure), oblique viewpoint (elliptical rings), and circular background clutter. Recommendations: add ellipse based rectification for oblique views and a negative image set to measure false accept rate.

5 Disc detection: generic preset driven path (`_detect_discs_generic`)

Purpose and outputs

- **Input**: preprocessed RGB image and `board_result`.
- **Output**: list of detected discs, each record containing centre, radius, quality/confidence, and a Lab colour vector.

High level strategy

1. Estimate a nominal disc radius r_0 from the detected centre radius
2. Compute board median brightness over the play area and select a detection preset via `_select_disc_params`. Presets set thresholds: `e_hit_min`, `sd_in_max`, `contrast_min`, `ring_margin`, `mid_std_max`, etc.
3. Enhance luminance (unsharp mask + CLAHE) to emphasize rims.
4. Run Hough circle detection over a narrow radius band around r_0.
5. For each candidate compute local descriptors and apply preset gating rules to accept/reject.
6. Apply non maximum suppression and remove centre hole artifacts.
7. Extract Lab patch means for accepted detections for later clustering.

Preset selection (example ranges)

- **bright_white**:

```
ring_margin = 0.25 * r0
e_hit_min = 0.28
sd_in_max = 0.15
contrast_min = 0.18
mid_std_max = 0.80
```

- **mid_wood** :

```
ring_margin ~ 0.3 * r0
e_hit_min ~ 0.10-0.25
sd_in_max ~ 0.22-0.25
contrast_min ~ 0.001-0.08
mid_std_max ~ 1.0-1.3
```

- **dark_or_shaded**:

```
ring_margin = 0.10 * r0
e_hit_min = 0.14
sd_in_max = 0.36
contrast_min = 0.01
mid_std_max = 1.7
```

Preprocessing

- Unsharp mask (radius=2, amount=1.5) to enhance rims.

- Convert to grayscale and apply CLAHE (clip_limit=0.01).
- Compute dual Canny edges (on luminance and inverted luminance) and combine.

Candidate descriptors For each Hough candidate:

- `_edge_ring`: mean edge strength sampled around circumference.
- `_inside_stats`: mean/std inside candidate and mean in a thin outer ring (contrast).
- `_angular_uniformity`: angular standard deviation at mid radius.
- Geometric checks: inside play mask and not inside centre hole.

Gating logic A candidate is accepted only if it satisfies the preset's conjunction of criteria:

- edge strength $\geq$ `e_hit_min`,
- interior std $\leq$ `sd_in_max`,
- contrast $\geq$ `contrast_min`,
- angular uniformity $\leq$ `mid_std_max`,
- and other preset dependent checks (delta from board median, not on forbidden ring margin unless strong evidence).

On ring candidates (centres near scoring ring radii within `ring_margin`) are suppressed unless strong evidence exists.

Post processing

- Non maximum suppression removes nearby duplicates (distance threshold approximately 0.9 times the larger radius).
- Remove any centres inside the centre hole and extract Lab patch means for clustering.

Failure modes and mitigations

- Overlapping discs: Hough may miss/merge overlapping rims. Mitigation: NMS helps some duplicates, recommended future work is watershed or instance segmentation.
- Ring edge ghosts: suppressed by onring logic and checks.
- Low contrast discs: presets tuned by brightness help, additional illumination correction or learned detectors may be needed.

6 Per disc scoring: `calculate_disc_scores`

Procedure

1. For each detected disc, sample $N = 32$ points uniformly around the nominal disc perimeter.
2. Query the scoring region mask at each sample point.
3. Base score = mode of sampled mask values.
4. If multiple scoring values are present among samples or any sample lies inside a dilated boundary band, assign the minimum of touched valid scores (lower of touched values rule).
5. If disc centre distance + radius < centre_radius - 1 px and base score is 15, upgrade to 20 (conservative fully in hole test).
6. Confidence = fraction of perimeter samples equal to the assigned score (reduced if line touch present).

Rationale Perimeter sampling implements the official rule deterministically and provides auditable sample points on overlays. The conservative fully in hole test avoids false 20 awards.

7 Small annotated code snippets

Below are short, annotated snippets taken and condensed from `Crokinole.py`. These are included to make the algorithm descriptions concrete and to show the key implementation patterns used in the pipeline.

Snippets are intentionally concise and annotated, they are not full functions but demonstrate the logic and important checks.

7.1 detect_board_and_rings (condensed, annotated)

```
# edges: binary Canny edge map
# config: contains ring_ratios and tolerances

# 1) Build adaptive radii array (target ~200 radii) then Hough:
hough_res = transform.hough_circle(edges, radii)
accums, cx, cy, radii_detected = transform.hough_circle_peaks(hough_res, radii,
                                                              total_num_peaks=120)

# 2) Turn peaks into candidate list [(x,y,r,accum), ...] and pick large candidates
ring5_candidates = [c for c in candidates if c.radius >= ring5_min]

# 3) For each ring5 candidate, search for inner rings by expected ratios:
for ring5 in ring5_candidates[:5]:
    board_radius = int(ring5.radius / ring_ratios['ring_5'])
    matches = {}
    for name, ratio in [('ring_10',0.66), ('ring_15',0.33), ('center',0.05)]:
        expected_r = board_radius * ratio
        radius_tol = max(5, expected_r * 0.15)
        center_tol = max(5, board_radius * 0.03)
        best = find_best_circle(cand, exp_r, ring5.center, radius_tol, center_tol)
        if best is None:
            break
        matches[name] = best
    if all inner rings matched:
        board_result = refine_board_center(ring5, matches)
        return board_result
# otherwise return None
```

Short explanation: the snippet shows Hough peak extraction, selection of outer ring candidates, and an inner ring search that uses combined radius and centre tolerances. If a coherent concentric set is found the board result is returned.

7.2 _detect_discs_generic (condensed, annotated)

```
# board_result provides ring centres/radii
r0, r_min, r_max = _expected_disc_radius(board_result, config)
params = _select_disc_params(board_brightness, r0)

# Preprocess to boost rims:
sharp = filters.unsharp_mask(img_float, radius=2, amount=1.5)
lum = exposure.equalize_adapthist(color.rgb2gray(sharp), clip_limit=0.01)

# Dual Canny to catch dark on light and light on dark rims
e1 = feature.canny(lum, sigma=1.5, low_threshold=low_t, high_threshold=high_t)
e2 = feature.canny(1.0 - lum, sigma=1.5, low_threshold=low_t, high_threshold=high_t)
edges = np.maximum(e1, e2)

# Hough over tight radii band:
hspaces = transform.hough_circle(edges, np.arange(r_min, r_max+1))
```

```
acc, hcx, hcy, rs = transform.hough_circle_peaks(hspaces, np.arange(r_min, r_max+1),
                                                 total_num_peaks=140)

candidates = []
for score, x, y, r in zip(acc, hcx, hcy, rs):
    if not play_mask[int(y), int(x)]: continue
    e_hit = _edge_ring(edges, x, y, r)
    mu_in, sd_in, mu_out = _inside_stats(lum, x, y, r)
    mid_std, mid_mean = _angular_uniformity(lum, x, y, r)
    contrast = abs(mu_in - mu_out)
    # gating according to preset:
    if e_hit < params['e_hit_min'] or
    sd_in > params['sd_in_max'] or contrast < params['contrast_min']:
        continue
    candidates.append((x, y, r, e_hit))

# Non max suppression and removal of centre hole artifacts:
picked = nms_by_distance_and_score(candidates)
# Extract Lab patches and return detections
```

Short explanation: this snippet shows the preprocessing, Hough candidate extraction, photometric descriptors, preset driven gating, and NMS that produce final disc detections.

7.3 calculate_disc_scores (condensed)

```
# For each detected disc (x,y,r), sample N points on perimeter
N = 32
angles = np.linspace(0, 2*np.pi, N, endpoint=False)
samples = []
for theta in angles:
    sx = int(x + r * np.cos(theta))
    sy = int(y + r * np.sin(theta))
    samples.append(scoring_mask[sy, sx])
# base score = mode(samples)
base = mode(samples)
# if multiple values touched or any sample in line_band -> assign min touched value
touched = set(samples) - {0}
if len(touched) > 1 or any(line_band[...]):
    assigned = min(touched) if touched else 0
else:
    assigned = base
# fully in hole upgrade:
if (distance_to_center + r) < (center_radius - 1) and assigned == 15:
    assigned = 20
confidence = samples.count(assigned) / float(N)
```

Short explanation: perimeter sampling reads the rasterized scoring mask, enforces the lower of touched values rule, performs the conservative in hole upgrade to 20, and computes a per disc confidence.

8 Evaluation methodology

The notebook driven evaluation writes `results/testset_results.csv` with columns:

`image_name, success, pred_team1, pred_team2, gt_team1, gt_team2,`

`abs_error_team1, abs_error_team2, error_pct_team1, error_pct_team2, num_discs, error_msg`

Reported metrics (team level):

- Exact match fraction for final totals.
- Per image absolute team errors and percent errors.
- Mean absolute final point error per image.
- Correlation of error with number of detected discs.

Limitations: per disc ground truth is not available, per disc detection metrics cannot be computed from the current repository contents.

9 Results

Per image team level results (verbatim from `results/testset_results.csv`):

Table 1: Per image summary (from `results/testset_results.csv`).

Image	Pred T1	Pred T2	GT T1	GT T2	AbsErr T1	AbsErr T2	Err% T1	Err% T2	NumDiscs
board1.jpg	30	30	30	30	0	0	0.00	0.00	4
board2.jpg	30	35	30	35	0	0	0.00	0.00	9
board3.jpg	55	50	55	50	0	0	0.00	0.00	15
board4.jpg	60	55	60	55	0	0	0.00	0.00	12
board5.jpg	20	100	20	100	0	0	0.00	0.00	13
ivan_board1.jpg	0	0	0	0	0	0	0.00	0.00	0
ivan_board2.jpg	25	20	25	20	0	0	0.00	0.00	4
ivan_board3.jpg	60	75	60	75	0	0	0.00	0.00	11
ivan_board4.jpg	15	25	15	25	0	0	0.00	0.00	4
ivan_board5.jpg	35	30	35	35	0	5	0.00	14.29	6
ivan_board6.jpg	30	35	30	35	0	0	0.00	0.00	6
ivan_board7.jpg	50	35	45	30	5	5	11.11	16.67	8
ivan_board8.jpg	65	75	65	75	0	0	0.00	0.00	14
sam_board1.jpg	0	0	0	0	0	0	0.00	0.00	0
sam_board2.jpg	10	10	10	10	0	0	0.00	0.00	2
sam_board3.jpg	25	10	25	10	0	0	0.00	0.00	4
sam_board4.jpg	5	15	5	15	0	0	0.00	0.00	4
sam_board5.jpg	20	25	20	25	0	0	0.00	0.00	6
sam_board6.jpg	60	60	50	60	10	0	20.00	0.00	12
sam_board7.jpg	15	75	20	85	5	10	25.00	11.76	10

Aggregate statistics

- Evaluated images: 20
- Exact final team totals: 16 / 20 = 80%
- Images with any non zero final error: 4 / 20 = 20%
- Total absolute final point error: 40 points
- Mean absolute final point error per image: 2.0 points
- Mean detected discs per image: 7.2

10 Representative overlays

Representative success: accurate detection and scoring (overlay).

Representative failure: ghost disc detection (false positives) and failure to detect discs (false negatives).

Figure 1: Representative overlay outputs.

11 Recommendations and future work

To improve robustness and address failure modes:

1. Perspective rectification (ellipse fitting + homography) to correct oblique views.
2. Overlap handling.
3. Collect per disc ground truth annotations for detection level evaluation.
4. Add an interactive review step in the notebooks to handle low confidence cases.

12 Conclusions

A deterministic, interpretable pipeline for Crokinole scoring has been implemented and evaluated. The pipeline attains exact team totals on the majority of evaluated images, remaining weaknesses are overlapping discs, near line sensitivity, and detection artifacts. The report documents algorithms, parameter choices, evaluation method and results, and provides prioritized recommendations.

Acknowledgements

Thanks to our peers Ivan and Sam for dataset assistance, and Dr. Mark Eramian for feedback.

A Appendix A: Individual contributions

- William Bushell: ring pattern detection, board detection, scoring mask creation.
- Rudra Patel: disc detection, disc automatic threshold, perimeter sampling scoring logic, and scoring calculation.
- Both: dataset collection, parameter tuning, testing, pipeline development and report preparation.

B Appendix B: Brief user manual

Environment

```
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Run evaluation

Open testset_script.ipynb and run all cells. The notebook calls run_complete_pipeline for each image and writes overlay PNGs and a summary CSV to results/.