BARRELS Scoring Engine: Complete Technical Breakdown

Last Updated: November 26, 2025

Table of Contents

1️⃣ Feature Extraction

Input Data Source

- **Single-camera pose estimation** using MediaPipe Pose (33 keypoints)
- Skeleton data format:

```typescript
interface SkeletonFrame {
frame: number;
timestamp: number; // milliseconds
keypoints: Keypoint[];
}

interface Keypoint {
x: number; // normalized 0-1 (frame width)
y: number; // normalized 0-1 (frame height)
z: number; // depth estimate (normalized, often unreliable)
visibility: number;// confidence 0-1
name: string; // e.g., “left_hip”, “right_wrist”
}
```

Coordinate System

- **Normalized coordinates**: All x, y values are in 0-1 range relative to frame dimensions
- **Origin**: Top-left corner (0,0), bottom-right (1,1)
- **Z-axis**: Depth estimate from camera (used cautiously due to single-camera limitations)
- **Conversion to physical**: Multiply by frame dimensions for pixel coordinates

Key Frames Detected

Auto-detection heuristics:

1. **Load Frame**: Frame with lowest hip Y-position (COM proxy)

- Heuristic: `min((leftHip.y + rightHip.y) / 2)` across all frames
- Buffer: Actual load = detected frame - 5 frames

1. **Launch Frame**: 10 frames after load (simplified)
   - Formula: `loadFrame + 10`
   - In production: Should detect when pelvis begins rotating

2. **Impact Frame**: User-specified or mid-swing default
   - Default: `Math.floor(skeletonData.length / 2)`
   - Ideally marked by coach/user during video review

Swing Phases

Not explicitly named as A/B/C, but implicitly tracked:
- **Phase A (Load)**: Stance → Load frame
- **Phase B (Launch)**: Load → Launch frame
- **Phase C (Swing)**: Launch → Impact frame

Joint-Based Features Extracted

🔹 MOTION Metrics (Velocity & Rotation)

1. Bat Speed (`lib/biomechanical-analysis.ts`)
- **Method**: Track lead wrist velocity in 5-frame window around impact
- **Formula**:

```typescript
  velocity = sqrt((x2 - x1)² + (y2 - y1)²) / (t2 - t1)
  batSpeed = avg(velocities[-3:])  // Last 3 frames before impact
```

- **Units**: Pixels/second → converted to mph via `* 0.682` (approximate calibration)
- **Normalization**: Assumes 60 fps baseline, ~180cm frame height

2. Angular Velocities (`lib/kinematic-sequence.ts`)
- **Segments tracked**:
- Pelvis: Angle of left_hip → right_hip line
- Torso: Angle of left_shoulder → right_shoulder line
- Arm: Angle of elbow → wrist line (lead arm)
- Bat: Approximated from arm angle (ideally tracked separately)

- **Formula** (Central Difference):

  ```typescript
    angularVelocity[i] = |angle[i+1] - angle[i-1]| / (2 * dt)
    // dt = 1/fps (time between frames)
  ```

- **Units**: Degrees per second

3. Hip Rotation (`lib/biomechanical-analysis.ts`)
- **Method**: Measure hip line rotation from stance to impact
- **Formula**:

```typescript
  initialAngle = atan2(rightHip.y - leftHip.y, rightHip.x - leftHip.x)
  impactAngle = atan2(rightHip.y - leftHip.y, rightHip.x - leftHip.x) at impact
  rotation = |impactAngle - initialAngle| * (180 / π)
```

- **Units**: Degrees
- **Range**: 0° (no rotation) to 90°+ (full turn through)

🔹 STABILITY Metrics (Angles & Positions)

4. Spine Tilt (`lib/swing-analyzer.ts`)
- **Method**: Angle of spine (hip midpoint → shoulder midpoint) from vertical
- **Formula**:

```typescript
  hipMid = midpoint(leftHip, rightHip)
  shoulderMid = midpoint(leftShoulder, rightShoulder)
  tilt = |atan2(shoulderMid.x - hipMid.x, shoulderMid.y - hipMid.y)| * (180 / π)
```

- **Units**: Degrees (0° = vertical, 90° = horizontal)
- **Measured at**: Launch frame and impact frame

5. Pelvis Angle
- **Method**: Angle of hip line from horizontal
- **Formula**:

```typescript
  angle = atan2(rightHip.y - leftHip.y, rightHip.x - leftHip.x) * (180 / π)
```

- **Measured at**: Launch and impact

6. Shoulder Tilt
- **Method**: Same as pelvis angle but for shoulders
- **Measured at**: Launch and impact

7. Knee Flexion (Back & Front)
- **Method**: 3-point angle at knee joint
- **Formula**:

```typescript
  kneeAngle = calculateAngle(hip, knee, ankle)
  // Uses dot product: cos(θ) = (BA · BC) / (|BA| * |BC|)
```

- **Units**: Degrees (180° = straight leg, 90° = deep bend)
- **Measured at**: Launch (back knee) and impact (front knee)

8. Elbow Angles (Lead & Rear)
- **Method**: 3-point angle at elbow joint
- **Formula**: Same as knee flexion, using (shoulder, elbow, wrist)
- **Measured at**: Impact frame

9. Hip-Shoulder Separation (X-Factor)
- **Method**: Hip rotation angle at launch (captured in `hipRotation.rotationAngle`)
- **Ideal**: 40-60° of separation

10. Head Displacement
- **Method**: Distance traveled by head from stance to impact
- **Formula**:

```typescript
  dx = impactHead.x - stanceHead.x
  dy = impactHead.y - stanceHead.y
  displacement = sqrt(dx² + dy²) * 180  // Scaled to ~cm
```

- **Units**: Approximate centimeters
- **Ideal**: < 10cm (“quiet head”)

11. Stride Length Factor
- **Method**: Front ankle displacement from stance to launch

- **Formula**:

```
typescript
  dx = launchAnkle.x - stanceAnkle.x
  dy = launchAnkle.y - stanceAnkle.y
  strideFactor = sqrt(dx² + dy²)  // Already normalized 0-1
```

- **Units**: Fraction of player height (0-1 range)
- **Ideal**: 0.4-0.6 (40-60% of height)

🔹 SEQUENCING Metrics (Timing)

12. Kinematic Sequence (`lib/kinematic-sequence.ts`)
- **Method**: Track peak angular velocity timing for 4 segments
- **Segments**: Pelvis → Torso → Arm → Bat
- **Output**:
- Peak velocity values (deg/s) for each segment
- Peak timing (ms before impact) for each segment
- Sequence order (e.g., `["pelvis", "torso", "arm", "bat"]`)
- Timing gaps between segments (ms)

13. Timing Metrics
- **Load to Launch**: `(launchFrame - loadFrame) * (1000 / fps)` ms
- **Launch to Impact**: `(impactFrame - launchFrame) * (1000 / fps)` ms
- **Total Swing Time**: Load to impact
- **Ideal ranges**:
- Load to Launch: 200-400ms
- Launch to Impact: 150-250ms
- Total: 400-600ms

2️⃣ Scoring Formulas

Overall Architecture

The scoring engine uses a **hierarchical weighted average** system:

```
Overall Score (0-100)
  ├─ Kinematic Sequence (40%) ← Dr. Kwon method
  ├─ Bat Speed (30%)          ← Normalized velocity
  ├─ Timing (15%)             ← Swing duration
  └─ Hip-Shoulder Sep (15%)   ← X-Factor
```

Per-Feature Scoring Functions

🔹 1. Kinematic Sequence Score (0-100)

Location: `lib/kinematic-sequence.ts:calculateSequenceScore()`

Formula:

```
sequenceScore = orderScore * 0.5 + timingScore * 0.5

// Order Score (50 points)
orderScore = order matches ["pelvis", "torso", "arm", "bat"] ? 50 : partialCredit
partialCredit = (correctPositions / 4) * 50

// Timing Score (50 points = 3 gaps × ~17 points each)
gap1 = |pelvisTiming - torsoTiming|
gap2 = |torsoTiming - armTiming|
gap3 = |armTiming - batTiming|

for each gap:
  if gap in [30-50ms]: gapScore = 100
  else: gapScore = max(0, 100 - |gap - 40ms| * 2)

timingScore = avg(gapScore1, gapScore2, gapScore3)
```

Ideal Pattern:
- Correct order: Pelvis peaks first, bat peaks last
- Timing gaps: 30-50ms between each segment (40ms ideal)
- **Perfect score**: 100 = correct order + all gaps in 30-50ms range

---

🔹 2. Bat Speed Score (0-100)

Location: `lib/swing-analyzer.ts:calculateOverallScore()`

Formula:

```
batSpeedMph = wristVelocity * 0.682  // pixel/s → mph conversion
batSpeedScore = min(100, max(0, ((batSpeedMph - 60) / 20) * 100))

// Linear scale:
// 60 mph or below = 0 points
// 80 mph or above = 100 points
// 70 mph = 50 points
```

Normalization:
- **Youth (8-12)**: 60-70 mph typical → 0-50 points
- **HS (13-18)**: 70-80 mph typical → 50-100 points
- **College**: 80-90 mph → often maxes out
- **Pro**: 90-100+ mph → maxes out at 100

Note: This is a placeholder. Production should use level-adjusted thresholds.

🔹 3. Timing Score (0-100)

Location: `lib/swing-analyzer.ts:calculateOverallScore()`

Formula:

```
idealTime = 500ms  // Launch to impact
timingScore = max(0, 100 - |totalSwingTime - idealTime| / 5)

// Every 5ms deviation = -1 point
// 400ms swing = 80 points
// 600ms swing = 80 points
// 500ms swing = 100 points
```

Ideal Pattern:
- Too fast (<400ms): Rushed, loss of control
- Too slow (>600ms): Late, poor timing

🔹 4. Hip-Shoulder Separation Score (0-100)

Location: `lib/swing-analyzer.ts:calculateOverallScore()`

Formula:

```
idealXFactor = 50°
xFactorScore = max(0, 100 - |xFactor - 50| * 2)

// Every 1° deviation = -2 points
// 40° or 60° = 80 points
// 50° = 100 points
// 30° or 70° = 60 points
```

Tolerance Band:
- **Excellent**: 45-55° (90-100 points)
- **Good**: 40-60° (80-100 points)
- **Needs Work**: <35° or >65° (<70 points)

Category Scores (Anchor, Engine, Whip)

Note: These are currently **mapped from existing database scores**, not calculated directly from joint features yet.

Location: `lib/engine-metrics-config.ts`

ANCHOR (Feet & Ground)

- **Motion** (40%): loadIntoBackLeg, strideMove, weightShift
- **Stability** (40%): headBalance, baseWidth, frontSideBrace
- **Sequencing** (20%): loadStrideTiming, groundUpStart, anchorSequence

Current Implementation:

```
// Mapped from existing assessment scores
ANCHOR = weightedAvg([
  anchorStance * 0.133,
  anchorMotion * 0.2,
  anchorWeightShift * 0.133,
  anchorStability * 0.4,
  anchorGroundConnection * 0.067,
  anchorSequencing * 0.2,
  anchorLowerBodyMechanics * 0.067
])
```

ENGINE (Hips & Shoulders)

- **Motion** (40%): hipTurn, shoulderTurn, hipShoulderStretch
- **Stability** (30%): postureControl, shoulderFinish, backLegSupport
- **Sequencing** (30%): hipsFirst, hipsToShouldersTiming, engineSequence

Current Implementation:

```
ENGINE = weightedAvg([
  engineHipRotation * 0.16,
  engineTorsoMechanics * 0.12,
  engineCorePower * 0.12,
  engineStability * 0.3,
  engineSequencing * 0.3
])
```

WHIP (Arms & Bat)

- **Motion** (40%): handPath, barrelTurn, releaseSpeed
- **Stability** (30%): contactPoint, barrelPlane, finishControl
- **Sequencing** (30%): engineToWhipTiming, handBreakLaunch, whipSequence

Current Implementation:

```
WHIP = weightedAvg([
  whipArmPath * 0.133,
  whipMotion * 0.2,
  whipBatSpeed * 0.133,
  whipStability * 0.3,
  whipSequencing * 0.167,
  whipConnection * 0.067
])
```

3️⃣ Baselines / Calibration

Reference Population

Current Status: No formal calibration dataset yet.

Intended Approach:
1. **Elite/Pro baseline**: MLB Statcast data (pelvis rotation, bat speed, sequence timing)
2. **Amateur benchmarks**: Collected from app users, segmented by level
3. **Normalization**: Level-specific percentile rankings

Level-Aware Normalization

Bat Speed Adjustment (Active)

Location: `lib/barrel-calculator.ts:getEvMinForLevel()`

```
function getEvMinForLevel(level: string): number {
  if (level === 'pro' || level === 'mlb') return 98;      // mph
  if (level === 'college' || level === 'hs') return 92;   // mph
  return 85;  // youth
}
```

Barrel Calculation:
- Maps player’s EV to MLB scale: `evForTable = evMph - (evMin - 98)`
- Example: HS 92 mph → MLB 98 mph equivalent
- Uses Statcast angle windows for barrel determination

Other Metrics (Planned)

Currently **NOT level-adjusted**:
- Hip rotation (should allow wider range for youth)
- Timing (youth swings are slower)
- X-Factor (smaller players, less separation)

Recommendation: Add level multipliers:

```
const LEVEL_MULTIPLIERS = {
  youth: { speed: 0.75, timing: 1.2, xFactor: 0.9 },
  hs: { speed: 0.9, timing: 1.1, xFactor: 0.95 },
  college: { speed: 1.0, timing: 1.0, xFactor: 1.0 },
  pro: { speed: 1.1, timing: 0.95, xFactor: 1.05 }
}
```

Outlier Corrections

Confidence Filtering:

```
function calculateConfidence(skeletonData): number {
  // Average joint visibility across all frames
  totalConfidence = sum(keypoint.visibility for all keypoints)
  return totalConfidence / keypointCount
}

// If confidence < 0.6: Flag as "Low Quality" but don't reject
```

Smoothing (Not currently applied):
- Suggested: Gaussian smoothing on angular velocity curves
- Reason: Single-camera pose can be jittery, especially in Z-axis

---

4️⃣ Final Score

BARREL Score Calculation

Formula (Weighted Average):

```
BARREL = (ANCHOR + ENGINE + WHIP) / 3

// Where each component is 0-100
```

Current Implementation (`lib/swing-analyzer.ts`):

```
overallScore = (
  sequenceScore * 0.40 +
  batSpeedScore * 0.30 +
  timingScore * 0.15 +
  xFactorScore * 0.15
) / weightSum

// If any component missing, adjust weightSum accordingly
```

Scale

- **Range**: 0-100 (percentage scale)
- **Display**: Integer (e.g., “85” not “85%”)
- **Grade Mapping**:
  `typescript`
  `  A: 90-100  (Elite)`
  `  B: 80-89   (Advanced)`
  `  C: 70-79   (Developing)`
  `  D: 60-69   (Beginner)`
  `  F: 0-59    (Needs Work)`

Combining Mechanics + Outputs

Current: Pure mechanics-based scoring
- No ball flight data (EV, LA, spin) integrated yet
- Skeleton/pose analysis only

Future: Hybrid scoring

```
// Proposed formula
FINAL_SCORE = (
  mechanicsScore * 0.6 +
  outputScore * 0.4
)

where:
  mechanicsScore = BARREL (Anchor + Engine + Whip)
  outputScore = barrelRate * 0.5 + avgEV * 0.3 + hardHitRate * 0.2
```

Why not 100% output-based?
- Young players can have great mechanics but lower output (strength)
- Output-only scoring doesn’t guide improvement
- Mechanics predict future output as player matures

5️⃣ Example Debug Output

Sample Swing Video Analysis

Input: 240fps video, 2.5 seconds (600 frames), right-handed hitter

Raw Feature Values

```
{
  "videoId": "vid_abc123",
  "frameCount": 600,
  "fps": 240,
  "impactFrame": 300,
  "loadFrame": 180,
  "launchFrame": 190,

  "rawFeatures": {
    "batSpeed": {
      "wristVelocity": 112.5,
      "batSpeedMph": 76.8,
      "maxSpeedMph": 78.2,
      "avgSpeedMph": 74.1,
      "unit": "mph"
    },

    "hipRotation": {
      "initialAngle": 5.2,
      "impactAngle": 48.7,
      "rotationDeg": 43.5,
      "peakRotationDeg": 51.3,
      "unit": "degrees"
    },

    "kinematicSequence": {
      "pelvis": {
        "maxVelocity": 627,
        "peakFrame": 275,
        "peakTimingMs": 104,
        "unit": "deg/s, ms before impact"
      },
      "torso": {
        "maxVelocity": 512,
        "peakFrame": 285,
        "peakTimingMs": 63,
        "unit": "deg/s, ms before impact"
      },
      "arm": {
        "maxVelocity": 890,
        "peakFrame": 295,
        "peakTimingMs": 21,
        "unit": "deg/s, ms before impact"
      },
      "bat": {
        "maxVelocity": 1240,
        "peakFrame": 300,
        "peakTimingMs": 0,
        "unit": "deg/s, ms before impact"
      },
      "sequenceOrder": ["pelvis", "torso", "arm", "bat"],
      "gaps": {
        "pelvisToTorso": 41,
        "torsoToArm": 42,
        "armToBat": 21,
        "unit": "ms"
      }
    },

    "angles": {
      "spineTiltLaunch": 12.3,
      "spineTiltImpact": 18.7,
      "pelvisAngleLaunch": 22.1,
      "pelvisAngleImpact": 48.3,
      "shoulderTiltLaunch": 8.9,
      "shoulderTiltImpact": 32.4,
      "frontKneeFlexionImpact": 168.2,
      "backKneeFlexionLaunch": 142.5,
      "leadElbowImpact": 135.7,
      "rearElbowImpact": 97.3,
      "unit": "degrees"
    },

    "stability": {
      "headDisplacementCm": 8.2,
      "strideLengthFactor": 0.52,
      "hipShoulderSeparation": 43.5,
      "unit": "cm, factor (0-1), degrees"
    },

    "timing": {
      "loadToLaunchMs": 42,
      "launchToImpactMs": 458,
      "totalSwingTimeMs": 500,
      "unit": "ms"
    }
  }
}
```

Per-Feature Scores

```
{
  "featureScores": {
    "batSpeed": {
      "rawValue": 76.8,
      "formula": "((76.8 - 60) / 20) * 100",
      "score": 84.0,
      "weight": 0.30,
      "contributionToTotal": 25.2,
      "grade": "B",
      "interpretation": "Above average bat speed for HS level"
    },

    "kinematicSequence": {
      "orderScore": 50,
      "orderReason": "Perfect sequence: pelvis → torso → arm → bat",

      "timingScores": {
        "pelvisToTorsoGap": {
          "gapMs": 41,
          "score": 98,
          "reason": "41ms is within ideal 30-50ms range"
        },
        "torsoToArmGap": {
          "gapMs": 42,
          "score": 96,
          "reason": "42ms is within ideal 30-50ms range"
        },
        "armToBatGap": {
          "gapMs": 21,
          "score": 62,
          "reason": "21ms is below ideal 30ms (too fast)"
        }
      },

      "avgTimingScore": 85.3,
      "finalSequenceScore": 92.7,
      "weight": 0.40,
      "contributionToTotal": 37.1,
      "grade": "A",
      "interpretation": "Excellent kinematic sequence with great timing"
    },

    "timing": {
      "rawValue": 500,
      "formula": "100 - |500 - 500| / 5",
      "score": 100,
      "weight": 0.15,
      "contributionToTotal": 15.0,
      "grade": "A",
      "interpretation": "Perfect swing tempo"
    },

    "xFactor": {
      "rawValue": 43.5,
      "formula": "100 - |43.5 - 50| * 2",
      "score": 87.0,
      "weight": 0.15,
      "contributionToTotal": 13.1,
      "grade": "B",
      "interpretation": "Good hip-shoulder separation"
    }
  }
}
```

Composite Score

```
{
  "compositeScore": {
    "method": "weighted_average",
    "totalWeight": 1.0,

    "components": [
      {
        "name": "kinematicSequence",
        "score": 92.7,
        "weight": 0.40,
        "contribution": 37.1
      },
      {
        "name": "batSpeed",
        "score": 84.0,
        "weight": 0.30,
        "contribution": 25.2
      },
      {
        "name": "timing",
        "score": 100,
        "weight": 0.15,
        "contribution": 15.0
      },
      {
        "name": "xFactor",
        "score": 87.0,
        "weight": 0.15,
        "contribution": 13.1
      }
    ],

    "rawScore": 90.4,
    "rounded": 90,
    "grade": "A",
    "percentile": null,
    "comment": "Percentile not yet calibrated - requires population data"
  }
}
```

Final Displayed Score

```json
{
  "finalScore": {
    "barrel": 90,
    "anchor": 88,
    "engine": 92,
    "whip": 89,

    "display": {
      "mainScore": 90,
      "mainLabel": "BARREL Score",
      "subScores": [
        { "label": "Anchor", "value": 88, "delta": -1 },
        { "label": "Engine", "value": 92, "delta": 2 },
        { "label": "Whip", "value": 89, "delta": 1 }
      ]
    },

    "interpretation": {
      "grade": "A",
      "level": "Elite",
      "summary": "Excellent swing mechanics with elite kinematic sequencing and perfect timing. Small improvement needed in hip-shoulder separation to reach 95+ score.",
      "strengths": [
        "Perfect kinematic sequence order",
        "Ideal swing tempo (500ms)",
        "Above-average bat speed for HS level",
        "Good timing gaps between segments"
      ],
      "improvements": [
        "Increase hip-shoulder separation to 45-50° (currently 43.5°)",
        "Arm-to-bat gap is too quick (21ms, target 30-40ms)",
        "Work on maintaining separation during load phase"
      ]
    }
  }
}
```

Adjustments & Penalties Applied

```
{
  "adjustments": {
    "confidencePenalty": {
      "applied": false,
      "reason": "Average joint visibility 0.87 (above 0.6 threshold)"
    },

    "levelAdjustment": {
      "applied": true,
      "playerLevel": "hs",
      "batSpeedThreshold": 92,
      "reason": "Bat speed normalized to HS level (92 mph min for barrels)"
    },

    "outlierCorrection": {
      "applied": false,
      "reason": "No extreme outliers detected in joint positions"
    },

    "smoothing": {
      "applied": false,
      "reason": "Not yet implemented"
    }
  },

  "bonuses": {
    "perfectSequence": {
      "applied": true,
      "bonus": 0,
      "reason": "Already counted in sequence order score"
    }
  }
}
```

Summary

What Works Now ✅

1. **Single-camera pose extraction** (33 keypoints, normalized coords)
2. **Auto-detection** of load/launch/impact frames (heuristic-based)
3. **Dr. Kwon kinematic sequencing** (pelvis→torso→arm→bat timing)
4. **Basic biomechanical metrics** (bat speed, hip rotation, angles, timing)
5. **Weighted composite scoring** (40% sequence, 30% speed, 15% timing, 15% x-factor)
6. **Level-adjusted barrel calculation** (youth/HS/pro thresholds)

What's Missing / Needs Work ⚠️

1. **No level-specific normalization** for most metrics (only bat speed)
2. **No population-based percentiles** (no calibration dataset)
3. **Anchor/Engine/Whip scores** currently mapped from legacy DB fields, not calculated from joints
4. **No ball flight integration** (exit velo, launch angle, barrel rate)
5. **No smoothing** on pose data (can be jittery)
6. **No multi-camera support** (depth/Z-axis unreliable)

7. **Swing phase detection** is very basic (should use pelvis rotation, not just frame count)
8. **No coach override** system for incorrect frame detection

Recommended Improvements 🎯

1. Add **level multipliers** for all metrics
2. Build **percentile lookup tables** from user data
3. Implement **direct joint→score mappings** for A/E/W (bypass legacy DB)
4. Integrate **HitTrax/Rapsodo output metrics** when available
5. Add **Gaussian smoothing** to angular velocity curves
6. Build **admin UI** for adjusting scoring weights and thresholds
7. Add **comparison mode**: player vs. model swing side-by-side scoring

---

End of Document