Implementation and Validation of an Action-Routed Seamless Loop Generation Plug-in for MotionBuilder

Xiongtao Nie s5804815

August 17, 2026

Abstract

Loop creation is a recurring task in game animation, virtual production, and motion-capture editing. A practical loop requires compatible start and end states, controlled root motion, stable foot contact, and a dependable route from analysis to export. This report examines the architecture, algorithms, deployment choices, and validation evidence of `seamless_loop_tool`, a Python plug-in for Autodesk MotionBuilder. The production path samples an 18-joint skeleton, constructs overlapping 45-frame windows, extracts a 175-dimensional motion descriptor, and uses a random forest to route a Take as walk, run, or other. Walk and run share one gait-loop detector; other or failed classifications require user confirmation. Candidate boundaries are generated from hips-trajectory peaks and bounded autocorrelation and are ranked with endpoint height, Euler rotation, local velocity, duration, and period terms. The processing stage supports in-place root conversion, full-interval linear endpoint-offset compensation, constant heading alignment, heuristic foot locking, and non-destructive sandbox-Take FBX export at 30, 60, 90, or 120 FPS. On 286 clips from a held-out subject, the current classifier produced 277 correct predictions, 96.85% accuracy, and a macro-F1 of 0.9532. Pure NumPy inference matches the training-side evaluator to a maximum probability error of 4.44×10^{-16}, with identical labels, and all 126 automated tests pass. The integrated design reduces host dependencies and makes unsupported motions recoverable through human confirmation. These results support implementation, routing, and code-verification claims, but they do not establish external-domain generalization or measured improvements in seam quality, foot sliding, latency, memory, or artist productivity. The report therefore separates verified current behavior from proposed future evaluation.

Keywords: MotionBuilder, Motion Capture, Seamless Loop, Action Classification, Random Forest, Root Motion, Foot Lock

1 Introduction

1.1 Production Problem

Looping turns a finite performance into a reusable animation asset. Repetition is essential for locomotion states in games, background characters in virtual production, previs blocking, simulation tests, and any interactive system that cannot store a unique performance for every possible duration. The editing operation appears simple because the final file contains only a finite interval. In practice, however, the first and last frames must describe compatible states of a moving articulated body. A visible seam can result from a mismatch in position, velocity, rotation, gait phase, or contact state. Even a small discrepancy becomes conspicuous when the same transition is repeated many times.

Root translation introduces a second production decision. A traveling loop preserves the displacement performed by the actor and is useful when a game or cinematic system expects authored root motion. An in-place loop removes horizontal travel so that locomotion can be driven by gameplay, navigation, or a separate trajectory controller. These two outputs share the need for a continuous pose, but they impose different constraints on the root. Moreover, endpoint agreement at the hips does not imply that a planted foot, swinging hand, or upper-body gesture is compatible. Loop preparation is therefore not one mathematical operation. It is a sequence of classification, candidate selection, root treatment, contact handling, plotting, inspection, and export.

Motion-graph research established the value of similarity-driven transitions between recorded motion segments, while interactive motion-data systems demonstrated how reusable clips can support responsive character animation (*Kovar Gleicher Pighin*, *2002*; *Lee et al.*, *2002*). Phase-aware control further shows that locomotion is organized by periodic state rather than by isolated frames (*Holden et al.*, *2017*). These ideas motivate the current system, but its production setting is narrower. The plug-in must work on the selected MotionBuilder Take, tolerate incomplete scene conditions, communicate uncertainty to an animator, and produce an editable result through the host application's own plotting and export mechanisms.

1.2 Project Objective and Scope

The goal of the current project is not to replace motion synthesis research with a new universal model. It is to provide an integrated MotionBuilder workflow that analyzes the

current Take, warns when a gait-specific detector may be inappropriate, constructs a loop, applies root and optional foot-contact processing, and exports the result without overwriting the source Take. MotionBuilder scene, time, plotting, Take, and export operations are accessed through the documented `pyfbsdk` boundary (*Autodesk*, *2026*). Core numerical routines remain separable from this boundary so that many behaviors can be tested without launching MotionBuilder.

The project contributes a compact action-routing layer, a reproducible motion descriptor, a portable random-forest evaluator, a root-based cycle detector, non-destructive application logic, and explicit handling of namespace and export-rate choices. Classification is deliberately advisory. Walk and run predictions permit automatic progression, whereas other or failed classifications produce a warning and allow a human override. This design preserves artist authority and prevents a statistical model from becoming an irreversible production gate.

The report focuses on what the current repository actually implements. It explains the information that passes between the UI, MotionBuilder adapter, classifier, loop-analysis routines, root processor, contact functions, and exporter. It also interprets the held-out classifier results and automated tests. It does not claim that the plug-in synthesizes novel motion, solves a global biomechanical optimization, or guarantees perceptually seamless output for every skeleton and action.

1.3 Contributions and Evidence Standard

Three forms of evidence are distinguished throughout the report. Implementation evidence shows that a feature is reachable in the production path. Verification evidence shows that covered behavior matches deterministic tests or stored validation artifacts. Effectiveness evidence requires a measured improvement in animation quality, runtime, or production work. Confusing these categories would make the evaluation appear stronger than it is. For example, the existence of foot-lock code proves an available correction mechanism, and unit tests may prove that contact intervals are mapped consistently, but neither fact establishes a numerical reduction in foot sliding on unseen production clips.

The current repository provides strong implementation and code-verification evidence, together with a subject-held-out validation of the routing classifier. It does not yet contain a controlled before-and-after animation study, an external capture-domain evaluation, or an animator productivity experiment. Accordingly, reported accuracy and macro-F1 apply to action routing on the stated validation set. Proposed seam, foot-contact, latency, memory, and usability metrics are presented as future evaluation criteria rather than completed results.

2 Related Work and Limitations

2.1 Motion Reuse, Transition Search, and Phase

Motion graphs formalize a useful principle for loop generation: transitions should be placed where two motion states are sufficiently compatible (*Kovar Gleicher Pighin*, *2002*). A graph system may search many clips and construct new paths through a motion database. The current plug-in uses the same general intuition at a smaller scale. It searches within one Take for two frames that can serve as a reusable interval. This narrower problem avoids graph construction and path planning, but it inherits the difficult question of how similarity should be defined. A root-only comparison is inexpensive and robust to joint-name variation, yet it can miss a disagreement in the extremities. A full-body comparison provides more information, but it depends on consistent skeleton resolution and an appropriate weighting of joints and rotations.

Interactive motion-data systems also show that recorded clips become more valuable when they can be reorganized into responsive assets (*Lee et al.*, *2002*). A MotionBuilder plug-in serves this production objective directly: it transforms an editing task that otherwise spans curve inspection, frame-range selection, root cleanup, plotting, and export into one connected workflow. Nevertheless, automation does not eliminate the need for an animator's judgment. Similarity scores are proxies for the perceptual result, and the most numerically similar boundary may not be the most useful loop for a particular design intent.

Locomotion is especially suited to periodic analysis because alternating support and swing phases generate repeated structure. Phase-functioned control demonstrates that phase can organize the relationship between pose and motion over a locomotion cycle (*Holden et al.*, *2017*). The current detector does not estimate a learned continuous phase variable. Instead, it uses peaks and autocorrelation of a root-derived signal to propose likely repetitions. This is simpler to deploy, but its success depends on the root trajectory expressing enough periodic variation. An in-place capture with very little horizontal movement or an unusual stylized gait can violate that assumption.

2.2 Boundary Continuity, Foot Contact, and Rotation

Boundary repair and foot-contact repair are related but different problems. Similar start and end root states do not guarantee that a planted foot remains fixed during support. Conversely, a locally stable foot does not guarantee continuity of the hips or upper body at the loop seam. Footskate-cleanup research therefore detects contact before modifying motion and treats support constraints as information that must be preserved during editing (*Kovar Schreiner Gleicher*, *2002*). The current plug-in follows the same broad detect-then-constrain pattern, but its implementation is a threshold-based FK/keying heuristic rather than a full-body optimization. It anchors horizontal position during detected contacts and clamps vertical position to the configured ground plane. It does not solve a coupled inverse-kinematics problem for the legs, hips, and center of mass.

Rotation representation is another source of boundary error. Quaternion curves provide spherical interpolation for three-dimensional orientation and avoid treating the components of a rotation as independent Euclidean coordinates (*Shoemake*, *1985*). Research on rotation representations further explains why compact parameterizations can have discontinuities that matter to learning and numerical comparison (*Zhou et al.*,

2019). The current endpoint cost compares Euler rotation channels because those channels correspond directly to the sampled root representation used by the loop detector. This choice is operationally simple, but wrap-around can make two nearly equivalent orientations appear far apart. The report therefore treats quaternion or geodesic endpoint comparison as a possible improvement, not as an existing capability.

2.3 Action Recognition and Deployment Trade-offs

Skeleton-action recognition provides context for the routing component. ST-GCN models the body as a spatiotemporal graph, 2s-AGCN adapts graph topology and combines joint and bone information, CTR-GCN refines channel-specific topology, and PoseConv3D represents skeleton sequences as three-dimensional heatmap volumes (*Chen et al.*, 2021; *Duan et al.*, 2022; *Shi et al.*, 2019; *Yan et al.*, 2018). These approaches demonstrate the value of learned spatial and temporal structure. They also target broader multi-class benchmarks and normally depend on deep-learning runtimes, model weights, and preprocessing stacks that are disproportionate for a three-way safety route inside a DCC host.

The plug-in instead uses a compact hand-designed descriptor and random forest. This sacrifices end-to-end representation learning but improves transparency and deployment control. Feature groups correspond to interpretable properties such as joint speed, displacement, root kinematics, energy, bilateral timing, and periodicity. Runtime evaluation needs NumPy rather than a complete training framework. A forest also supports a direct JSON representation in which schema, thresholds, class order, and tree nodes can be validated before inference. The choice is therefore justified by the application boundary rather than by a claim that random forests are generally superior to graph or convolutional networks.

2.4 Datasets, Generalization, and Privacy

The selected motion corpus is based on MPI HDM05 represented through AMASS and routed with BABEL labels. AMASS provides a common body-model representation for heterogeneous motion-capture sources, BABEL supplies temporally aligned semantic action labels, and the HDM05 documentation records source actions and subjects (*Mahmood et al.*, 2019; *Müller et al.*, 2007; *Punnakkal et al.*, 2021). This combination makes it possible to construct training windows with a consistent skeleton representation while retaining action semantics.

Large skeleton datasets emphasize that evaluation protocols must separate identities and acquisition conditions (*Liu et al.*, 2020; *Shahroudy et al.*, 2016). The current validation separates a held-out subject, which is stronger than a random window split because adjacent windows from the same performance cannot appear on both sides of the evaluation. It still does not constitute a second external dataset. Studio layout, marker processing, retargeting, body proportion, frame rate, and action style may all change the feature distribution. Finally, skeleton trajectories may preserve identity-related information; motion data should not automatically be treated as anonymous merely because texture and facial appearance are absent (*Moon et al.*, 2023). Dataset governance and consent remain relevant if the tool is extended beyond the current research corpus.

3 System Architecture and Action Routing

3.1 Component Boundaries

The plug-in separates user interaction, MotionBuilder adaptation, motion classification, loop analysis, root processing, contact handling, and export. This division is important because only part of the system can run outside MotionBuilder. The UI collects options and displays state. The adapter translates requests into `pyfbsdk` scene, time, model, Take, plotting, and file operations. Core modules operate on NumPy arrays and versioned model data. As a result, feature extraction, forest traversal, route selection, candidate scoring, root correction, and several time-mapping behaviors can be verified in an ordinary Python test environment.

The boundary also limits the effect of a host-application failure. A missing SDK, unavailable model, invalid character, short Take, unresolved joint, or damaged classifier file can be represented as a diagnostic outcome instead of an uncontrolled numerical result. The production workflow does not assume that every scene is fully characterized or that every naming convention matches the defaults. This matters in a DCC tool because scene state is mutable and often contains several characters, constraints, imported namespaces, and temporary Takes.

3.2 Decision Flow and Human Override

Clicking `Analyze` first invokes the classifier. A walk or run decision proceeds to loop analysis. An other or failed decision presents a confirmation message, and the user may still continue. Classification therefore acts as a reversible safety layer rather than a hard gate. The route answers a narrow question: whether the selected Take resembles the locomotion types for which the gait-cycle search was designed. It does not label every frame or determine the artistic value of the motion.

The confirmation path is an intentional risk-control mechanism. A false walk or run prediction can send non-periodic motion to a detector whose assumptions are inappropriate, while a false other prediction can unnecessarily interrupt a valid gait workflow. Allowing an override makes both errors recoverable. The reason field returned by classification also distinguishes a threshold decision from a failure condition, so the interface can communicate why automatic progression was not selected.

Walk and run currently call the same gait-loop detector. The repository does not contain separate production implementations named `walk_loop_detector` and `run_loop_detector`, and it does not apply class-specific scoring weights. The class label changes the safety decision, not the mathematical ranking of loop candidates. This distinction prevents the architecture diagram from implying two specialized algorithms that do not exist.

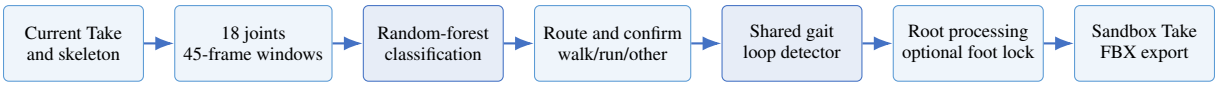

Figure 1. Implemented production path. Walk and run share the same loop detector; other or failed classifications require confirmation.

3.3 Scene Resolution and State Safety

Namespace resolution prefers exact names and long names. If a short name maps to more than one scene object, the system raises an ambiguity error rather than silently selecting a skeleton. The UI exposes a namespace field and can derive a namespace from a selected bone, then applies the normalized prefix to the configured hips, feet, and toe names. These safeguards are necessary in multi-character scenes because an incorrect root or joint binding invalidates classification, candidate generation, contact detection, and export at the same time.

The workflow also separates analysis from application. Analysis identifies a proposed interval without rewriting the source. Processing stores the motion and contact information needed for the chosen options. Application writes to the intended target, and export creates a sandbox Take with a collision-safe name. This staged structure makes user choices visible and avoids treating the current scene as a disposable buffer. It also supports a degraded test adapter when the MotionBuilder SDK is unavailable, which is useful for verifying orchestration behavior without claiming that host-specific plotting has been exercised.

Table 1. Routing and Classifier Configuration.

Item	Current Value	Implemented Meaning
Walk route	Threshold 0.50	Continue to the shared gait detector
Run route	Threshold 0.48	Continue to the shared gait detector
Other/failure	Confirmation	Warn and allow an explicit override
Forest	400 trees	Maximum depth 8; random seed 42
Sampling	0.8 samples	Maximum feature fraction 0.5
Class weights	Balanced	`balanced_subsample`

4 Motion Representation and Classification

4.1 Sampling and Canonical Skeleton Input

The runtime sampler resolves 18 standard body locations: hips; bilateral upper legs, legs, feet, and toe bases; spine, neck, and head; and bilateral arms, forearms, and hands. The set covers lower-body periodicity, root travel, trunk response, and upper-body activity without requiring every optional joint in a detailed production rig. Rotations are represented as global 3×3 matrices for descriptor extraction. Root translation is converted to metres, which aligns the runtime representation with the training data and prevents centimetre-valued MotionBuilder positions from shifting speed-related features by two orders of magnitude.

Input motion is designed around 30 FPS. The classifier constructs 45-frame windows with a stride of 15 frames, so each window represents approximately 1.5 seconds and adjacent windows overlap by one second. The duration is long enough to contain meaningful locomotion structure while the stride provides multiple observations for a longer Take. A Take that is shorter than the required window cannot produce a compatible descriptor and is routed through a safe failure outcome rather than padded with invented motion.

4.2 Descriptor Construction

Each window becomes a 175-dimensional descriptor. The first 72 dimensions summarize joint angular speed with mean, standard deviation, 90th percentile, and maximum values for all 18 joints. The next 54 summarize angular excursion from the first-frame orientation using mean, standard deviation, and maximum. These statistics retain information about movement intensity and range while removing dependence on the exact starting pose.

Nineteen root features describe horizontal speed percentiles, vertical velocity, horizontal acceleration, vertical range, path length, net displacement, and trajectory straightness. Twelve energy features aggregate squared angular speed over lower-body, trunk, upper-body, and full-body groups. Fourteen symmetry features measure the peak correlation and normalized lag for seven left–right joint pairs. The final four periodicity features summarize autocorrelation strength and normalized period for lower-body and full-body energy. Collectively, the groups express local swing, overall travel, intensity, bilateral timing, and repetition. They are not a lossless encoding of the animation; they are a compact decision representation for routing.

The feature schema is versioned and includes joint order, frame count, descriptor size, selected SMPL indices, feature names, and a SHA-256 digest. Runtime loading rejects a model when its schema hash or descriptor size disagrees with the extractor. This protects against a subtle deployment error in which a valid numeric vector is interpreted using the wrong feature order.

4.3 Forest Selection and Portable Inference

The estimator is a random forest (*Breiman*, *2001*). Training uses scikit-learn for fitting, group-stratified five-fold validation, and 30 randomized configurations (*Pedregosa et al.*, *2011*). The selected configuration contains 400 trees, maximum depth 8, maximum feature fraction 0.5, maximum sample fraction 0.8, minimum leaf size 4, and minimum split size 10. Balanced subsampling addresses unequal class frequency during tree construction. The selected grouped cross-validation macro-F1 is 0.8660.

The deployed JSON stores tree structures, class order, walk

and run thresholds, schema information, deployment status, training metadata, and file integrity information. Runtime inference traverses the exported trees with NumPy rather than importing scikit-learn (*Harris et al.*, *2020*). Loading validates the model format, classes, probability thresholds, node-array shapes, child indices, feature indices, probability normalization, reachable nodes, deployment status, and optional checksum. This is more than a serialization convenience: it reduces the dependency surface inside MotionBuilder and turns compatibility failures into explicit messages.

Parity testing compares portable inference with the training-side evaluator. The maximum absolute probability difference is 4.44×10^{-16}, and all parity labels are identical. This establishes numerical equivalence for the evaluated samples; it does not establish classifier accuracy on a new motion domain.

4.4 Clip-Level Routing

Probabilities are predicted for every window and averaged before one route is chosen for the complete Take. Walk passes when its mean probability is at least 0.50 and greater than run. Run passes when its mean probability is at least 0.48 and greater than walk. All remaining cases fall back to other. This conservative rule requires a dedicated class to be both sufficiently confident and more plausible than the competing gait class.

Clip-level aggregation is suitable for deciding whether a Take is mainly gait or non-gait motion, but it cannot localize several actions in a long sequence. A Take containing idle, run, and jump phases may be routed according to whichever phase dominates its windows. Averaging can also hide a short high-confidence gait segment inside mostly unrelated motion. Temporal segmentation, transition detection, or window-level visualization would be separate future features and are not implied by the current classifier.

Table 2. Training and Validation Composition.

Split	Subjects or Classes	Count
Training	bk 787; mm 547; dg 524	1,858 windows
Training class	Other 965; walk 631; run 262	1,858 windows
Validation	Subject tr	2,840 windows
Validation clips	Other 168; walk 100; run 18	286 clips
Other detail	Mixed 137; combat 15; jump 8; idle 7; dance 1	168 clips

5 Loop Candidate Generation and Scoring

5.1 Trajectory Signal and Candidate Proposal

Loop analysis receives a six-channel root trajectory $[X, Y, Z, \mathrm{RotX}, \mathrm{RotY}, \mathrm{RotZ}]$. The current gait detector uses the vertical hips channel as its periodic signal. Local maxima are the recommended candidates because they can correspond to a repeatable part of the locomotion rhythm; valleys remain available in the core analyzer. Peak separation defaults to half of the configured minimum cycle length, which prevents neighbouring samples around one broad maximum from being treated as distinct phases.

Candidate restriction is both a modeling decision and a practical optimization. Comparing every possible pair in a Take would include many combinations that are too short, too long, or obviously out of phase. Peaks convert the search into pairs of plausible phase landmarks. Pair duration must meet the configured minimum and may not exceed 2.5 times the configured maximum. The detector also estimates a likely lag by subtracting the mean of the hips signal and evaluating autocorrelation over a bounded range. The selected lag is treated as a half-period, while twice that value represents a preferred full left–right stride.

If fewer than two candidates survive, the current fallback is $(0, N/2)$. The fallback makes the system deterministic and allows the workflow to continue, but it is not a quality-optimal estimate. It should be interpreted as a recoverable low-information result that requires inspection, not as evidence that a valid gait cycle was detected.

5.2 Endpoint and Velocity Cost

For candidate endpoints (i) and (j), the implemented cost can be summarized as

$$C(i,j) = 2|Y_i - Y_j| + \|\mathbf{R}_i - \mathbf{R}_j\|_2 + 5\|\mathbf{V}_i - \mathbf{V}_j\|_2 + B(i,j), \tag{1}$$

where $\mathbf{R}$ contains Euler root rotations, $\mathbf{V}$ is local velocity over all six root channels, and B is a period bonus. A full-period match within 10 frames receives -0.3; a half-period match within 5 frames receives -0.1. Mean horizontal speed is converted with a fixed 30 FPS factor and compared with the default threshold 5. This can reject valid in-place gait whose root does not travel horizontally.

The vertical endpoint difference receives an explicit factor of two, while the Euclidean difference of root Euler channels supplies the remaining position term. Velocity is calculated from the next sample at each endpoint, and its mismatch receives a weight of five. This reflects the observation that a loop may match position yet still pop because the body enters and leaves the seam in different directions. A minimum vertical-bounce parameter is also available, although its default is zero. Invalid indices, reversed ranges, insufficient travel, or an inadequate segment return an infinite cost so that they cannot win the ranking.

The period bonus is deliberately smaller than the principal mismatch terms. It breaks ties in favour of a complete alternating stride without allowing duration alone to override a poor endpoint. The use of fixed frame tolerances and a fixed 30 FPS velocity conversion is nevertheless a limitation. If the incoming motion has not been normalized to the assumed rate, the same physical movement can produce a different numerical speed or duration decision.

5.3 Selection Scope and Failure Modes

The repository contains a helper for full-body pose difference, and tests cover that helper, but the production candidate-

ranking path does not call it. The selected loop is therefore the best root-endpoint match under the current cost, not the best whole-body pose match. Arms, feet, or the torso may still be out of phase when root states are similar. Contact state is also absent from the ranking, so two root peaks can be paired even when the supporting leg differs.

These limits explain why candidate selection and endpoint correction remain separate stages. Ranking chooses a promising interval; later processing reduces its root offset and optionally constrains detected contacts. Neither stage can retroactively prove that the original boundary was perceptually optimal. A stronger future ranking could combine geodesic joint rotation, relative joint position, root velocity, and left/right contact phase. Such an extension would require an explicit weighting and evaluation protocol, because adding more terms does not automatically produce a better perceptual metric.

6 Root Motion, Foot Contact, and MotionBuilder Export

6.1 In-Place and Traveling Root Treatment

In-place conversion sets root X and Z translations to zero while preserving Y and all three rotation channels. This keeps vertical body bounce and facing motion while delegating ground-plane displacement to an external controller. Setting the horizontal channels to zero, rather than merely subtracting the first frame, produces a root that remains at the world origin across the interval. The result is appropriate for an in-place asset but intentionally discards the actor's authored travel path.

For traveling output, endpoint offset is distributed linearly across the complete interval. If $\mathbf{p}_t$ is a root position and T is the number of samples, the conceptual correction is

$$\widetilde{\mathbf{p}}_t = \mathbf{p}_t - \frac{t}{T-1}\left(\mathbf{p}_{T-1} - \mathbf{p}_0\right). \tag{2}$$

The formula leaves the first sample unchanged and applies the complete endpoint correction at the last sample. Intermediate samples receive a proportional fraction. In the implementation, the sign is expressed through the first-minus-last delta, which is algebraically equivalent to the equation above. The correction avoids concentrating the entire discontinuity at the seam. For root-motion output, the forward channel can be excluded so that closing the loop does not erase intended travel or create a return teleport.

Heading alignment applies a constant RotY offset based on the difference between the requested target heading and the first root orientation. Applying one offset to the entire interval preserves relative yaw changes. The current `blend_frames` argument does not create a local crossfade window; it remains for API compatibility while compensation is distributed over the full interval. Linear compensation removes a direct endpoint difference but does not guarantee matching acceleration, angular velocity, or higher-order continuity.

6.2 Contact Detection and Stance Correction

Foot contacts are detected from the original motion during `Process`. Default thresholds are height 2.0, speed 0.5, and a minimum span of three frames. During `Apply`, contacts are mapped from source to target FPS, world-space X/Z is anchored, and Y is clamped to ground height. This heuristic can reduce obvious sliding, but the repository has no before-and-after sole-velocity measurement, full-body IK optimization, or knee-flip benchmark.

Detection combines the configured foot and optional toe height with a motion metric derived from the sampled trajectory. Consecutive samples that meet the height and speed conditions are converted into intervals only when they satisfy the minimum span. The intervals are stored in absolute source-frame coordinates together with the detected source FPS. Storing them during processing is important: application may occur after resampling or after other scene operations, and recomputing contact from already corrected motion could change the decision.

When the target rate differs, interval boundaries are mapped through time rather than copied as equal frame numbers. During a contact, the horizontal lock point is taken from the beginning of the interval and the vertical coordinate is set to the ground plane. The approach is predictable and inexpensive, but it does not distribute compensating motion through the knee, hips, or spine. A long or incorrectly detected interval can therefore over-constrain the foot, and a short noisy interval can leave visible sliding. Foot and toe names, ground height, thresholds, and axis conventions must correspond to the actual rig.

6.3 Non-Destructive Plotting and Export

Export uses a separate sandbox Take so that the source is not overwritten. The output Take uses the source name plus `_inplace` and a numeric suffix on collision. Users can select 30, 60, 90, or 120 FPS. The adapter sets the transport rate, plots on frames, and exports only the target Take through the MotionBuilder API (*Autodesk*, *2026*).

The sandbox is copied from the current Take and becomes the active destination for processing. Unique-name generation prevents a pre-existing output Take from being replaced merely because it uses the expected suffix. Plot options disable plotting across every Take, request frame-aligned sampling, set the target period, and avoid a constant-key reducer that could silently change the baked result. FBX options discard elements by default and then select the named target Take for export.

This sequence reduces destructive risk but does not make export transactional in the database sense. A host exception, invalid file path, missing current character, or unavailable SDK can still interrupt the operation. The source Take remains a recovery point, and explicit errors are preferable to silently exporting an unintended Take. Large-scene performance, constraint compatibility, retargeted control-rig behavior, and round-trip results in downstream engines remain subjects for a dedicated compatibility study.

Table 3. Implemented Processing and Evidence Limits.

Area	Implemented	Not Established
In-place	Zero X/Z; keep Y and rotation	Recovery of original travel speed
Endpoint	Full-interval linear offset	Local blend window or higher-order continuity
Heading	Constant RotY offset	Quaternion global optimization
Foot lock	Contact threshold and X/Z anchor	Measured slide reduction or full-body IK
Export	Sandbox Take and four FPS choices	Large-scene timing and compatibility matrix

7 Validation Results

7.1 Evaluation Protocol

Validation is organized at clip level because the production decision is made after averaging window probabilities for a complete Take. Windows from the held-out performer are processed by the same descriptor and portable routing rules used at runtime, then grouped by source clip. This prevents a long clip with many overlapping windows from being counted as many independent production decisions. It also keeps threshold evaluation aligned with the user-visible outcome: one route and one reason per Take.

Model selection and threshold calibration are distinct from final held-out evaluation. Training uses subject groups during cross-validation, and walk/run thresholds are selected from out-of-fold training probabilities aggregated by clip. The held-out performer is not used to choose the forest or revise the thresholds. This separation reduces direct leakage from the reported validation clips into model selection, although reuse of the same source corpus still limits conclusions about external domains.

The held-out subject contributes 286 clips. With true rows and predicted columns ordered as walk, run, and other, the confusion matrix is

$$\mathbf{M} = \begin{bmatrix} 95 & 1 & 4 \\ 0 & 17 & 1 \\ 2 & 1 & 165 \end{bmatrix}. \tag{3}$$

7.2 Metric and Error Interpretation

The model produces 277 correct predictions, 96.85% accuracy, and macro-F1 0.9532. Macro averaging assigns equal importance to the three routing classes rather than allowing the 168 other clips to dominate the summary. Walk accounts for five errors: one clip is routed as run and four as other. Run accounts for one error routed as other. Among other clips, two are routed as walk and one as run. These three false gait routes are operationally important because they can advance an unsuitable Take toward the gait detector, although the subsequent workflow remains inspectable and reversible.

The run estimate is the least stable because only 18 run clips are available. One additional error would noticeably change its precision or recall. High accuracy therefore should not be interpreted as equal certainty for every class or movement style. The result reflects the selected subject, label mapping, clip aggregation, and conservative thresholds. Calibration curves, confidence intervals, and a larger run set would provide a better account of uncertainty than additional decimal places in one point estimate.

Table 4. Clip-Level Validation Metrics.

Class	Precision	Recall	F1
Walk	0.9794	0.9500	0.9645
Run	0.8947	0.9444	0.9189
Other	0.9706	0.9821	0.9763
Macro	0.9482	0.9589	0.9532

7.3 Software Verification Evidence

The complete automated suite currently reports 126 passed tests. It covers feature modes, schema and model state, thresholds, NumPy parity, route decisions, MotionBuilder sampling substitutes, namespaces, candidate generation and fallback behavior, root processing, orientation alignment, FPS-aware contact mapping, sandbox Takes, export options, UI orchestration, and degraded operation without the SDK. The tests are fast because host-independent logic uses arrays and adapters rather than requiring MotionBuilder for every case.

Passing tests support code-level correctness only for their specified inputs. They do not demonstrate that a plotted foot looks natural, that an FBX behaves identically in every downstream engine, or that an animator prefers the generated boundary. Host integration also includes API and scene-state behaviors that a test double cannot completely reproduce. A fixed collection of MotionBuilder scenes and exported reference files would complement the current suite with end-to-end regression evidence.

Model selection used grouped cross-validation to reduce subject leakage. Dependent resampling and repeated model comparison can still produce optimistic conclusions, so the held-out results should remain separate from tuning evidence (*Dietterich*, *1998*). The grouped cross-validation macro-F1 of 0.8660 and the held-out clip macro-F1 of 0.9532 use different aggregation units and should not be described as an 8.7-point improvement.

8 Limitations and Industrial Meaning

8.1 Validity Threats

The first limitation is domain coverage. Training and validation both originate from routed MPI HDM05 material, and the run class has limited support. Subject separation does not automatically represent a new capture studio, character proportion, retargeting method, frame rate, naming convention, or movement style. A credible external study should add source-level separation and report calibration and rejection behavior, not only accuracy.

The second limitation is time normalization. The classifier window was designed for 30 FPS. Input Takes at other rates can change the physical duration of a 45-frame window

unless motion is resampled before feature extraction. The loop detector's horizontal-speed conversion also assumes 30 FPS. Explicit time normalization and a gait gate based on leg periodicity would improve in-place handling.

The third limitation is animation-quality evidence. Candidate ranking does not use active whole-body pose or foot-contact phase. Linear root compensation does not enforce higher-order continuity, and foot locking has no quantitative comparison. The next evaluation should measure root and joint position/velocity jumps at the seam, support-foot horizontal travel, knee-flip rate, analysis/apply/export latency, and peak memory. An animator study could then measure completion time, manual edits, and blinded quality ratings.

Construct validity is equally important. Classifier accuracy measures whether a routing label matches the dataset label; it does not measure seamlessness. Endpoint equality measures one aspect of a boundary but may not predict a viewer's response to acceleration, balance, or contact artifacts. A future experiment must define each outcome separately and avoid combining unrelated improvements into one unsupported quality score.

8.2 Production Interpretation and Next Evaluation

Despite these limits, the project has clear production value. Classification, confirmation, loop analysis, non-destructive processing, and export are connected in one workflow. Model schema and hash checks reduce deployment mismatch, and pure NumPy inference avoids a training-framework dependency. The random forest is not intended to compete with broad skeleton-action leaderboards; it is a compact decision layer for a constrained MotionBuilder task. A future model change should be driven by observed external failure modes rather than model popularity.

The most useful next study would use a locked set of traveling and in-place clips spanning body proportions, capture sources, frame rates, and retargeting conditions. For every clip, the study should record the selected boundary, root and joint discontinuity before and after processing, support-foot displacement, processing time, export success, and any manual correction. A blinded animator review could score loop acceptability without revealing whether the boundary was generated automatically. These measurements would convert current implementation evidence into effectiveness evidence while preserving failure cases for regression testing.

Deployment should also document prerequisites and recovery paths. Characterization, namespace selection, root and foot naming, ground height, source rate, target rate, and output location affect the result. The sandbox Take protects the source, but users still need to review the plotted output before treating an FBX as final. Clear warnings and recoverable state are therefore as important to industrial usefulness as a small improvement in one average score.

Table 5. Evidence Scope and Next Measurements.

Evidence	Current Result	Next Measurement
Classification	96.85% accuracy; 0.9532 macro-F1	Cross-source calibration and rejection
Inference	4.44×10^{-16} maximum parity difference	MotionBuilder load and inference time
Tests	126/126 passed	Fixed production-scene regression set
Seam quality	Not yet quantified	Position, rotation, and velocity jumps
Foot contact	Heuristic implemented	Support-foot travel and artifact rate
Productivity	Not yet quantified	Time, edits, and blinded animator rating

9 Conclusion

The current project is accurately described as an action-routed seamless-loop generation plug-in for MotionBuilder. It implements three-class motion routing, a shared gait-loop detector, root-motion processing, optional heuristic foot locking, namespace safeguards, sandbox Takes, and multi-rate FBX export. The classifier reaches 96.85% accuracy and 0.9532 macro-F1 on 286 held-out-subject clips, its NumPy evaluator matches the training-side implementation, and all 126 automated tests pass.

The evidence supports integration and classifier-routing claims but not universal animation-quality claims. The production path does not contain separate walk and run loop detectors, does not use full-body pose ranking, and has not measured seam reduction, foot-slide improvement, latency, memory, or productivity. The most valuable next milestone is therefore a reproducible MotionBuilder benchmark that measures boundary continuity and support-foot motion before and after processing.

The central engineering contribution is the connection of several modest, inspectable mechanisms into a recoverable host workflow. A versioned descriptor and portable forest determine when gait analysis is plausible; root peaks and autocorrelation restrict candidate search; endpoint and velocity terms rank intervals; root, heading, and contact operations prepare an output; and a uniquely named sandbox Take protects the original animation. Human confirmation remains available when the statistical route is uncertain or inappropriate.

Future work should prioritize evidence before algorithmic complexity. External-source validation would test the routing model, while a scene-based benchmark would test seam continuity, stance stability, processing cost, and export reliability. Only after those failure modes are measured should the project decide whether it needs class-specific detectors, contact-aware ranking, quaternion costs, full-body optimization, or a different classifier. In its current form, the plug-in is a validated integration prototype with a strong routing result and explicit production safeguards, not a universal solution to motion-loop generation.

References

Autodesk. (2026). *Python scripting for MotionBuilder: Reference guide (pyfbsdk)*. Autodesk Help.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010933404324

Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., & Hu, W. (2021). Channel-wise topology refinement graph convolution for skeleton-based action recognition. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13359–13368. https://openaccess.thecvf.com/content/ICCV2021/html/Chen_Channel-Wise_Topology_Refinement_Graph_Convolution_for_Skeleton-Based_Action_Recognition_ICCV_2021_paper.html

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

Duan, H., Zhao, Y., Chen, K., Lin, D., & Dai, B. (2022). Revisiting skeleton-based action recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2969–2978. https://openaccess.thecvf.com/content/CVPR2022/html/Duan_Revisiting_Skeleton-Based_Action_Recognition_CVPR_2022_paper.html

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Holden, D., Komura, T., & Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, *36*(4), 42:1–42:13. https://doi.org/10.1145/3072959.3073663

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, *21*(3), 473–482. https://doi.org/10.1145/566654.566605

Kovar, L., Schreiner, J., & Gleicher, M. (2002). Footskate cleanup for motion capture editing. *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 97–104. https://doi.org/10.1145/545261.545277

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., & Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, *21*(3), 491–500. https://doi.org/10.1145/566654.566607

Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.-Y., & Kot, A. C. (2020). NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(10), 2684–2701. https://doi.org/10.1109/TPAMI.2019.2916873

Mahmood, N., Ghorbani, N., Troje, N. F., Pons-Moll, G., & Black, M. J. (2019). AMASS: Archive of motion capture as surface shapes. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5442–5451. https://openaccess.thecvf.com/content_ICCV_2019/html/Mahmood_AMASS_Archive_of_Motion_Capture_As_Surface_Shapes_ICCV_2019_paper.html

Moon, S., Kim, M., Qin, Z., Liu, Y., & Kim, D. (2023). Anonymization for skeleton action recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(12), 15028–15036. https://doi.org/10.1609/aaai.v37i12.26754

Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., & Weber, A. (2007). *Documentation mocap database HDM05* (tech. rep. No. CG-2007-2). Universität Bonn. https://researchgate.net/publication/231521391_Documentation_Mocap_database_HDM05

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

Punnakkal, A. R., Chandrasekaran, A., Athanasiou, N., Quiros-Ramirez, A., & Black, M. J. (2021). BABEL: Bodies, action and behavior with english labels. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 722–731. https://openaccess.thecvf.com/content/CVPR2021/html/Punnakkal_BABEL_Bodies_Action_and_Behavior_With_English_Labels_CVPR_2021_paper.html

Shahroudy, A., Liu, J., Ng, T.-T., & Wang, G. (2016). NTU RGB+D: A large scale dataset for 3D human activity analysis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1010–1019. https://doi.org/10.1109/CVPR.2016.115

Shi, L., Zhang, Y., Cheng, J., & Lu, H. (2019). Two-stream adaptive graph convolutional networks for skeleton-based action recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12026–12035. https://openaccess.thecvf.com/content_CVPR_2019/html/Shi_Two-Stream_Adaptive_Graph_Convolutional_Networks_for_Skeleton-Based_Action_Recognition_CVPR_2019_paper.html

Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, *19*(3), 245–254. https://doi.org/10.1145/325334.325242

Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1), 7444–7452. https://doi.org/10.1609/aaai.v32i1.12328

Zhou, Y., Barnes, C., Lu, J., Yang, J., & Li, H. (2019). On the continuity of rotation representations in neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5745–5753. https://doi.org/10.1109/CVPR.2019.00589