CHAPTER III

Theoretical Background of G2O

G2O (General Graph Optimization) is a versatile and efficient open-source framework designed to solve nonlinear least squares problems that frequently occur in computer vision and robotics. G2O (General Graph Optimization) is a state-of-the-art C++ framework designed for solving nonlinear least squares problems commonly encountered in computer vision and robotics. It is widely used for tasks like SLAM (Simultaneous Localization and Mapping), Bundle Adjustment, and pose graph optimization.

In our case, we apply G2O to optimize 2D human keypoints under temporal and anatomical constraints.

G2O was introduced by Rainer Kuemmerle et al. in their 2011 paper:
"**g²o: A General Framework for Graph Optimization**"
(Kuemmerle et al., 2011)

In occluded frames, OpenPose may miss keypoints or produce noisy results. By modeling temporal and structural relationships in a graph, G2O propagates information from confident frames (non-occluded) to uncertain ones (occluded), filling gaps and smoothing noise.

3.1 Core Optimization Formulation in G2O

G2O solves nonlinear optimization problems that can be expressed as the minimization of a sum of squared error terms. This is formally defined as:

$$\min_{\mathbf{x}} \sum_{i=1}^{m} \|\mathbf{e}_i(\mathbf{x})\|^2_{\mathbf{\Omega}_i} = \min_{\mathbf{x}} \sum_{i=1}^{m} \mathbf{e}_i(\mathbf{x})^\top \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

Where:

- x: The vector of parameters to be optimized. In our case, this may include 2D keypoint coordinates across video frames.
- $e_i(x)$: The residual error function for the *i*-th constraint or measurement, representing the deviation between the estimated and observed values.
- Ω_i: The information matrix, which is the inverse of the covariance matrix associated with the measurement. It defines the confidence or weight of each error term — higher values indicate more reliable constraints.

3.2 Graph-Based Formulation of Optimization Problems

In G2O, the optimization problem is represented as a factor graph or pose graph, which encodes both the variables to be estimated and their interdependencies. The graph is denoted as:

$$\mathbf{G = (V, E)}$$

Where:

- V is the set of vertices, with each vertex representing a variable to be optimized. This could be, for example, a robot pose, a camera parameter, or — in our case — a 2D human keypoint.
- E is the set of edges, where each edge represents a measurement, observation, or constraint between variables.

Each edge $e_i(x)$is associated with an error function that measures the deviation between the observed measurement and the estimated value based on the current variables. The goal of optimization is to find the set of variables x that minimizes the total error induced by all such constraints.

This graph-based formulation allows G2O to model complex systems with many interrelated components and to exploit sparsity in the problem structure for computational efficiency.

g2o provides a modular framework with:

1. Vertices (Nodes): Represent variables (e.g., VertexSE2 for 2D poses).
2. Edges (Constraints): Define error functions (e.g., EdgeSE2 for pose-pose constraints).
3. Solvers: Handle linear system solving (e.g., BlockSolver, LinearSolver).
4. Optimization Algorithms: Gauss-Newton, Levenberg-Marquardt, etc.

3.3 G20 Optimization

G2O addresses nonlinear least-squares problems using efficient first-order iterative methods. The process involves several core steps, as outlined below.

3.3.1 Optimization Methods

G2O primarily uses the following optimization techniques:

1. Gauss-Newton Method:
   Suitable for well-conditioned problems where the Jacobian is accurate. It linearizes the error function and solves the resulting system iteratively.

2. Levenberg–Marquardt Algorithm:
   A damped version of Gauss-Newton that improves robustness for poor initial guesses or ill-conditioned systems by introducing a regularization term.

3.3.2 Iterative Optimization Process

The optimization process involves the following steps:

1. Linearization
   Each nonlinear error function e_i(x) is linearized using a first-order Taylor expansion:
2. System Construction
   The linearized equations are assembled into a global sparse linear system:
3. Sparse Solving
   Efficient sparse solvers such as Cholesky decomposition are used to solve for Δx, the update step.
4. Update Step
   The estimated parameters are updated iteratively as:

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$$

The steps are repeated until convergence criteria are met.

3.3.3 Benefits and Applicability

G2O is particularly advantageous when:

- The problem includes a large number of variables and constraints.
- The structure of the problem is sparse, meaning each constraint affects only a few variables.
- There is a need for custom-defined vertices and edges.

G2O also supports incremental updates and is suitable for real-time optimization tasks such as SLAM, 3D reconstruction, and pose refinement.

3.3.4 Application: Human Keypoint Optimization

In this research, G2O is applied to optimize 2D human keypoints obtained from OpenPose. The setup includes:

- Vertices: Represent 2D keypoints across video frames.
- Edges: Encode constraints to guide optimization:
    - Temporal Constraints: Encourage smooth transitions between frames.
    - Anatomical Constraints: Maintain consistent limb lengths.
    - Data Fidelity Constraints: Trust OpenPose predictions with high confidence.

By minimizing the total energy in the graph, G2O produces more stable, accurate, and anatomically consistent keypoint trajectories, especially in cases of occlusion, motion blur, or low-confidence detections.

Chapter 4

AT-G2O Algorithm

Anatomical-Temporal Pose Refinement using Graph Optimization

in this paper, we introduce **AT-G2O** (*Anatomical-Temporal Pose Refinement using Graph Optimization*), a novel algorithm designed to refine 2D human pose estimations by integrating anatomical and temporal constraints within a graph optimization framework. While existing 2D pose estimation methods often suffer from inconsistencies due to occlusions, noise, or missing data, AT-G2O addresses these challenges by leveraging the structural relationships inherent in the human skeleton and the temporal continuity of motion sequences.

The core of AT-G2O lies in constructing a graph where each node represents a joint position, and edges encode both anatomical connections (e.g., bones) and temporal links across consecutive frames. By formulating pose refinement as a graph optimization problem, we employ the g²o framework to minimize discrepancies while adhering to anatomical plausibility and motion smoothness.

Our approach operates in two stages: initially fixing high-confidence joints to stabilize the optimization, followed by a refinement phase that relaxes these constraints to fine-tune all joint positions. This two-tiered strategy ensures robustness against initial estimation errors and enhances the overall coherence of the pose sequences.

Through extensive experiments, we demonstrate that AT-G2O significantly improves the accuracy and consistency of 2D pose estimations, making it a valuable tool for applications in human-computer interaction, animation, and biomechanics.

In the AT-G2O framework, each human pose is represented by a fixed set of joints The joints indices and their corresponding anatomical names used in the skeleton model are as follows: Index 0 corresponds to the Neck, 1 to the Nose, 2 to the Right Shoulder, 3 to the Right Elbow, 4 to the Right Wrist, 5 to the Left Shoulder, 6 to the Left Elbow, 7 to the Left Wrist, 8 to the Right Hip, 9 to the Right Knee, 10 to the Right Ankle, 11 to the Left Hip, 12 to the Left Knee, and 13 to the Left Ankle. The connections between these joints are based on standard anatomical relationships, facilitating accurate pose estimation and analysis. This structure serves as the foundation for applying anatomical constraints in our optimization algorithm.

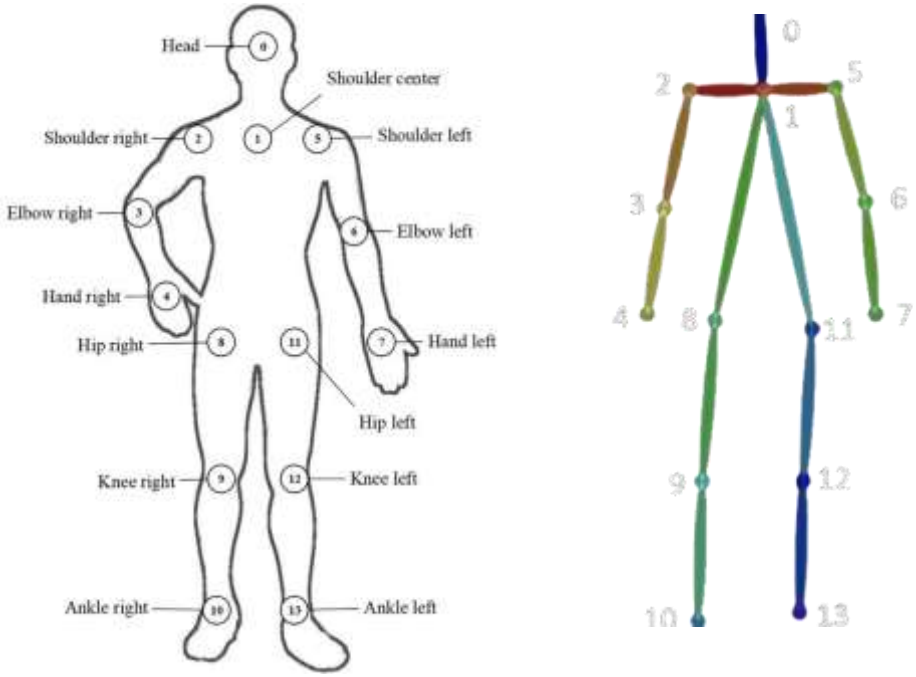

Figure 4.1.*Anatomical Skeleton Structure with Defined Joint Connections.*
The figure illustrates the key body joints with their corresponding indices: 0 (Neck), 1 (Nose), 2 (Right Shoulder), 3 (Right Elbow), 4 (Right Wrist), 5 (Left Shoulder), 6 (Left Elbow), 7 (Left Wrist), 8 (Right Hip), 9 (Right Knee), 10 (Right Ankle), 11 (Left Hip), 12 (Left Knee), and 13 (Left Ankle).

Our algorithm is as shown in Algorithm 1.

Algorithm 1: *Anatomical-Temporal Pose Refinement using Graph Optimization*

Input: OpenPose keypoint JSON files for multiple persons and sequences (extracted from video).

Output: Optimized keypoint JSON files for each sequence (excluding the reference).

Step 1: For Each Person Folder (p1, p2, p3)
1.1 Find all sequence folders (e.g., p11, p12, p13)
1.2 For each sequence folder:
Load all keypoint JSONs into a sequence array
Filter out low-confidence keypoints (set to NaN if confidence < 0.3)
Interpolate missing/low-confidence keypoints using linear interpolation
1.3 Set the first sequence (pX1) as reference:
Compute reference bone lengths (median length for each skeleton connection)

Step 2 : For Each Non-Reference Sequence
2.1 Initialize a g2o optimizer
2.2 For each frame:
Create a vertex for each keypoint (fixed if confidence > 0.5)
2.3 Add temporal edges:
For each keypoint, connect consecutive frames
Use reference sequence movement and confidence to set edge weights
2.4 Add skeleton (anatomical) constraints:
For each skeleton connection, add an edge to maintain reference bone length (soft constraint)
2.5 Run two-stage optimization:
Stage 1, Initial optimization (10 iterations)
Stage 2, Refinement (20 iterations)
2.6 Extract optimized keypoints from the optimizer
2.7 Apply temporal smoothing (Savitzky-Golay filter) to reduce jitter
2.8 Save optimized keypoints as new JSON files in an output folder

Step 3: Repeat for all persons and sequences

In step 1 Per-Person Keypoint Preprocessing stage operates on raw OpenPose outputs prior to optimization. Each OpenPose output contains 2D pose estimations per frame. These outputs must be organized and refined to ensure robust, temporally consistent keypoint sequences for each subject across multiple sequences. The refinement process improves keypoint quality by removing low-confidence detections and estimating missing joints using interpolation. Additionally, canonical bone lengths are computed for later constraint-based optimization.

1.1 Identifying Sequence Folders

Each dataset typically stores the OpenPose results in a hierarchical folder structure. At the top level, each folder represents one subject (e.g., p1, p2, p3), and inside each subject folder, there are multiple sequence folders containing OpenPose outputs in JSON format.

1.1.1 Subject Folder

Let:

$$\mathbf{P \in \{p1,p2,\ldots,pn\}}$$

represent the set of subject folders, where each P corresponds to a unique person. Each subject folder contains multiple subfolders representing different recording sessions (e.g., different actions, angles, or times).

1.1.2 Sequence Folder

For a given subject P, the folder contains a set of sequences:

$$\mathbf{P=\{S1,S2,\ldots,Sm\}}$$

where Si is the folder for the i^{th} sequence of that subject.

Each sequence folder S_i contains T OpenPose JSON files:

$$\mathbf{S_i=\{F1,F2,\ldots,F_T\}}$$

where each Ft contains the detected keypoints of frame t in that sequence.

Each JSON file FtF_tFt includes:

- 2D joint locations $(x_{t,j}, y_{t,j}) \in R^2$
- Confidence score $c_{t,j} \in [0,1]$
- For all joints j=1,2……J

Thus, the keypoint vector for joint j in frame t of sequence s_i is:

$$K^{(i)} = [K^i_{t,j}]_{t=1,\ldots,T;\ j=1,\ldots,J}$$

This structured data serves as the input for confidence filtering and interpolation, detailed in subsequent steps.

1.2 Processing Each Sequence Folder

In this phase, for each action sequence of a person, we extract frame-wise keypoint detections, denoise the data by filtering unreliable estimates, and interpolate missing values to construct temporally continuous 2D trajectories.

This step ensures that the optimization algorithm receives smooth, confidence-aware input trajectories per keypoint, thereby improving convergence and anatomical plausibility.

1.2.1 Loading JSON Keypoints into a Sequence Tensor

Each folder Si corresponds to a motion sequence and contains a set of frame-wise OpenPose JSON files:

$$S_i = \{F1, F2, \ldots, F_T\}$$

where T is the number of frames in the sequence, and each FtF_tFt is a JSON file containing:

- A list of J body keypoints for each person detected.
- Each keypoint $j \in \{1,2,...,J\}$ is given by:

$$k_{t,j} = (x_{t,j}, y_{t,j}, c_{t,j})$$

where:

- $X_{t,j}, y_{t,j} \in R$:2D pixel coordinates of joint jjj in frame ttt.
- $c_{t,j} \in [0,1]$: Confidence score of the detection.

These detections are collected into a tensor for the full sequence:

$$K^{(i)} = [K^{(i)}_{t,j}]_{t=1,\ldots,T;\ j=1,\ldots,J} \in R^{T \times J \times 3}$$

This tensor forms the raw spatiotemporal representation of keypoints for that sequence.

1.2.2 Confidence-Based Keypoint Filtering

To reduce noise, low-confidence keypoints (typically arising from occlusion, truncation, or poor lighting) are invalidated using a thresholding rule. This operation increases the robustness of downstream processing.

Let $\tau=0.3$ be the confidence threshold. The spatial components of keypoint j in frame t are set to NaN if:

$$C^{i}_{t,j} < \tau \Rightarrow x^{i}_{t,j} = \mathbf{NaN} \quad , \quad y^{i}_{t,j} = \mathbf{NaN}$$

1.2.3 Temporal Interpolation of Missing Keypoints

To fill missing or low-confidence joints and ensure continuity across frames, we use **linear interpolation** per joint over the time axis.

Let jjj be a specific joint. Define $T_j \subset \{1,2,\ldots,T\}$ as the set of frames where joint j was confidently detected (i.e., not NaN).

For missing values at frame $t \notin T_{jt}$ estimate the position using its two neighboring valid frames

Edge Cases:

- If missing values occur at the start or end of a sequence, linear interpolation is not possible.
- In such cases, either:
    - Apply **zero-order hold** (copy nearest known value),
    - Use **spline extrapolation**, or
    - **Drop the sequence** if too many frames are missing (e.g., >30%).

1.2.4 Result: Dense and Smoothed Keypoint Trajectories

Following the filtering and interpolation process, the output is a refined keypoint trajectory matrix denoted as:

$$\hat{\mathbf{K}}^{(i)} = [\ \hat{x}_{t,j}^{(i)}\ ,\ \hat{y}_{t,j}^{(i)}\ ] \in \mathbb{R}^{TxJx2}$$

These trajectories are temporally consistent, confidence-aware, and suitable for anatomical constraints and temporal optimization steps in later stages.

Temporal Consistency

The trajectories are temporally consistent due to the interpolation filling in missing or unreliable keypoints across frames. This continuity reduces abrupt spatial jumps and preserves the natural motion dynamics of the subject. Such temporal smoothness is crucial because human movement typically changes gradually rather than abruptly frame-to-frame, especially in high frame-rate video sequences.

Confidence Awareness

The filtering step ensures that only keypoints with sufficient confidence contribute to the interpolation and final trajectory. Low-confidence detections are excluded or treated as missing data, so the refined trajectories better reflect the true anatomical positions rather than noisy measurements. This confidence-aware approach prevents propagation of errors caused by occlusion, motion blur, or misdetections, improving the reliability of the pose data.

Suitability for Anatomical Constraints and Optimization

The refined trajectories $K^{(i)}$ form a clean and dense dataset that is well-suited for downstream processing steps involving anatomical constraints and temporal optimization, including:

Bone length constraints**:** Enforcing consistent lengths between connected joints over time, based on human anatomy, which stabilizes poses and removes implausible deformations.

Temporal smoothing and regularization**:** Applying optimization algorithms that further reduce noise while preserving motion fidelity.

Pose refinement**:** Integrating geometric or probabilistic models (such as graph-based optimizers) to correct remaining errors.

Because the trajectories are dense (no missing data) and smoothed, these optimization techniques can effectively leverage temporal and spatial correlations to produce accurate and physically plausible pose sequences.

1.3 Set the First Sequence as Reference

To ensure anatomical consistency across all pose sequences, a reference skeleton is computed from the first available sequence of each subject. This reference will be used to enforce bone-length constraints during optimization.

1.3.1 Choosing the Reference Sequence

Let $P \in \{p_1, p_2, ..., p_n\}$ be a subject folder.
Define the first sequence folder $S_1 \in P$ (typically named ‘pX1’) as the reference sequence for that

subject.
Let the interpolated and refined keypoints from Section 1.2 for this sequence be:

$$K(1) = [\hat{x}_{\{t,j\}}, \hat{y}_{\{t,j\}}] \in \mathbb{R}^{\{T \times J \times 2\}}$$

where:
• T is the number of frames in the sequence
• J is the number of joints
• $\hat{x}_{\{t,j\}}$, $\hat{y}_{\{t,j\}}$ are the interpolated 2D coordinates of joint j at frame t

1.3.2 Defining Skeleton Connections

Let $\mathfrak{B} = \{ (j_a, j\_b) \} \subseteq \{1, 2, ..., J\} \times \{1, 2, ..., J\}$ denote the set of bone connections, where each pair (j_a, j_b) represents a bone between two joints.

Example: $\mathfrak{B}$ = { (neck, right_shoulder), (hip, knee), ... }

1.3.3 Computing Bone Lengths Per Frame

In pose estimation, the human body is represented as a set of keypoints (joints) connected by bones. Each bone connects two joints, say joint A and joint B.

For each frame in a video or sequence, the position of each joint is estimated as a coordinate in 2D space (typically, pixel coordinates in the image). To compute the length of a particular bone in that frame, we calculate the straight-line distance between the two joint positions it connects.

This distance is known as the Euclidean distance, which is the shortest path connecting two points in a plane. It is computed by finding the difference between the horizontal (x) coordinates of the two joints, the difference between their vertical (y) coordinates, and then combining these differences using the Pythagorean theorem. The result is a single scalar value representing the bone length for that frame.

By repeating this calculation for every bone in the skeleton and every frame in the sequence, we obtain a time series of bone lengths that describe how each bone's apparent length changes over time.

1.3.4 Median Bone Length Over Time

Because the bone lengths can vary slightly due to noise, inaccuracies in joint detection, or movement, directly using bone lengths from individual frames might lead to instability.

To address this, we compute a reference bone length for each bone by taking the median of all the bone lengths calculated across all frames. The median is a robust statistical measure that is less sensitive to outliers or sudden fluctuations compared to the average.

This median length represents a typical or "central" bone length value over the entire sequence. It acts as a stable estimate of the true anatomical length of that bone for the subject in the video.

Collecting these median lengths for all bones gives us a reference skeleton with fixed bone lengths that reflect the subject's natural proportions.

Step 2: Optimization of Non-Reference Sequences

2.1 Initialize the Optimizer

2.1.1 Purpose of the Optimizer
The optimizer serves as the computational engine that minimizes the total error in a pose graph. A pose graph is a mathematical structure that encodes relationships (edges) between elements (vertices). In our case:
- Vertices represent keypoint locations for each frame.
- Edges represent spatial and temporal relationships (such as motion smoothness or anatomical constraints).
The optimizer will adjust keypoint positions across all frames to minimize a global energy function.

2.1.2 Optimizer Structure
Mathematically, we aim to solve a nonlinear least squares problem:

$$\mathbf{min}_{v} \sum_{i} ||f_i(v)||^2$$

Where:
- v represents all keypoint positions across all

- f_i(v) is the residual function for constraint i.
- The sum is over all constraints in the graph (temporal, anatomical, etc.).
This formulation allows us to optimize the entire motion trajectory while respecting structural and dynamic properties.

2.2 Vertex Creation for Each Keypoint and Frame

2.2.1 Keypoint Representation
Each keypoint at a specific frame is modeled as a vertex $V_{\{t,k\}} \in \mathbb{R}^2$ (for 2D) or $\mathbb{R}^3$ (for 3D), where:
- t is the time frame.
- k is the index of the keypoint (e.g., nose, elbow, etc.).

2.2.2 Confidence-Based Fixing
To leverage reliable observations, we fix the vertex if its confidence exceeds a certain threshold (commonly 0.5):

$$\mathbf{Fixed_{t,k}=\{\ true\ if\ c_{t,k}>0.5\ ||\ otherwise\ false\ \}}$$

This prevents the optimizer from altering trustworthy keypoints while allowing uncertain ones to move during optimization.

2.3 Temporal Constraints (Temporal Edges)

2.3.1 Purpose
Temporal constraints ensure that the same keypoint moves consistently across consecutive frames. They are essential for producing smooth motion and correcting noise or missing data.

2.3.2 Formulation
For each keypoint k and for each pair of consecutive frames (t, t+1), we define a residual based on the expected motion from the reference sequence
This encourages each keypoint in the non-reference sequence to follow the same motion pattern as the reference, if confidence is high.

2.4 Anatomical Constraints (Skeleton Edges)

2.4.1 Purpose
Skeleton (or anatomical) constraints maintain the relative distances between keypoints that are physically connected on the human body (e.g., shoulder to elbow). These help preserve the realism and biological plausibility of the motion.

2.4.2 Reference Bone Length

The reference bone length defines the expected distance between two anatomically connected keypoints, such as joints connected by a bone. It is derived from a reliable reference sequence of poses, which captures the typical proportions of the subject's body.

For each pair of connected keypoints, the reference bone length represents the typical distance between them in the reference data. This value acts as a standard or target for maintaining anatomical consistency during pose optimization.

By using these reference lengths, the optimization process can constrain the estimated poses so that the relative distances between connected keypoints remain consistent with natural human anatomy. This helps prevent unrealistic distortions like stretching or compressing of limbs, thus improving the quality and plausibility of the reconstructed poses.

2.4.3 Skeleton Constraint Formulation

The skeleton constraint ensures that the spatial relationships between keypoints connected anatomically (such as joints linked by bones) remain consistent during optimization. This is crucial because human body parts are connected by bones of relatively fixed length, and large deviations would produce anatomically impossible poses.

Anatomical Bone Connections (S):
The human skeleton is modeled as a set S of pairs of keypoints (ki,kj), where each pair represents two connected joints (for example, shoulder to elbow, elbow to wrist). These connections define the "bones" of the skeleton.

Reference Bone Lengths:
Each bone is associated with a reference length derived from a trusted pose sequence (the reference sequence). This length corresponds to the distance between the two connected keypoints in the reference pose, assumed to be anatomically accurate.

2.5 Two-Stage Optimization

2.5.1 Purpose
A two-stage process ensures both convergence and fine refinement. The first stage provides a coarse alignment, while the second stage corrects residual errors.

2.5.2 Stage 1 – Initial Optimization
A small number of iterations (e.g., 10) are used to quickly reduce large errors. The optimizer resolves gross inconsistencies using fixed and confident points as anchors.

2.5.3 Stage 2 – Refinement
A larger number of iterations (e.g., 20) are applied to refine details and ensure optimal convergence. During this stage, small positional adjustments are made based on accumulated residuals.

2.6 Extract Optimized Keypoints

2.6.1 Purpose
After optimization, we need to retrieve the refined positions of all keypoints across all frames.

2.6.2 Mathematical Form
Each optimized keypoint is:

$$\hat{\mathbf{v}}_{\{t,k\}} = \mathbf{argmin}_{\{v\{t,k\}\}} \left(E_{\mathbf{temporal}} + E_{\mathbf{skeleton}}\right)$$

This value is taken directly from the optimized vertices in the graph.

2.7 Apply Temporal Smoothing (Savitzky–Golay Filter)

2.7.1 Purpose
Even after optimization, small frame-to-frame jitter may remain. A temporal filter smooths the final trajectories to improve visual quality and realism.

2.7.2 Savitzky–Golay Smoothing
Apply the filter to each keypoint trajectory $\hat{v}\_\{:,k\}$ (across time):

$$\hat{\mathbf{v}}_{\{:,k\}}^{\text{smooth}} = \text{SGFilter}(\hat{\mathbf{v}}_{\{:,k\}}, \text{window\_length, polyorder})$$

Typical parameters:
- Window length: 7 frames
- Polynomial order: 3

This filtering is applied independently to each coordinate (x, y, or x, y, z).

2.8 Save Optimized Keypoints

2.8.1 Purpose

The final step stores the smoothed, optimized keypoints for further analysis or use in animation, visualization, or machine learning.

2.8.2 Output Format

Each frame's keypoints are stored in a structured format (e.g., JSON), preserving:
- Frame index
- Keypoint positions (x, y, or x, y, z)
- Confidence scores (optional, may be updated or kept original)

This step enables integration with existing pose analysis or visualization pipelines.

Step 3: Repeat for All Persons and Sequences

In multi-person, multi-sequence pose estimation scenarios, the optimization process must be applied consistently across all individual subjects and all recorded sequences to ensure comprehensive and coherent results.

3.1 Multiple Persons:

Each person detected in the video or dataset represents an independent set of keypoints and corresponding skeleton structure. The optimization process must be repeated separately for each individual to accurately refine their unique pose trajectories. This ensures that the temporal and anatomical constraints are tailored to each person's movement patterns, body proportions, and confidence levels in the keypoint detections.

3.2 Multiple Sequences:

Often, pose data is collected in multiple sequences or takes, each representing different actions, camera views, or time segments. To achieve robust and accurate pose estimation across diverse conditions, the entire optimization pipeline — including initialization, temporal smoothing, and constraint enforcement — must be applied independently to each sequence. This prevents contamination of data across different contexts and respects the unique dynamics and temporal correlations within each sequence.

3.3 Consistency and Scalability:

Repeating the optimization for all persons and sequences ensures that the approach scales to complex datasets involving multiple actors and diverse motion patterns. It maintains consistency

in pose refinement and allows for batch processing, which is critical in large-scale applications like sports analytics, animation, or behavioral studies.

3.4 Output Integration:
After processing all persons and sequences independently, their optimized keypoints can be collected, analyzed, or visualized collectively. This comprehensive approach enables downstream tasks such as multi-person interaction analysis, motion comparison, or further machine learning applications on high-quality, temporally consistent pose data.

.

4.1 **Flowchart**

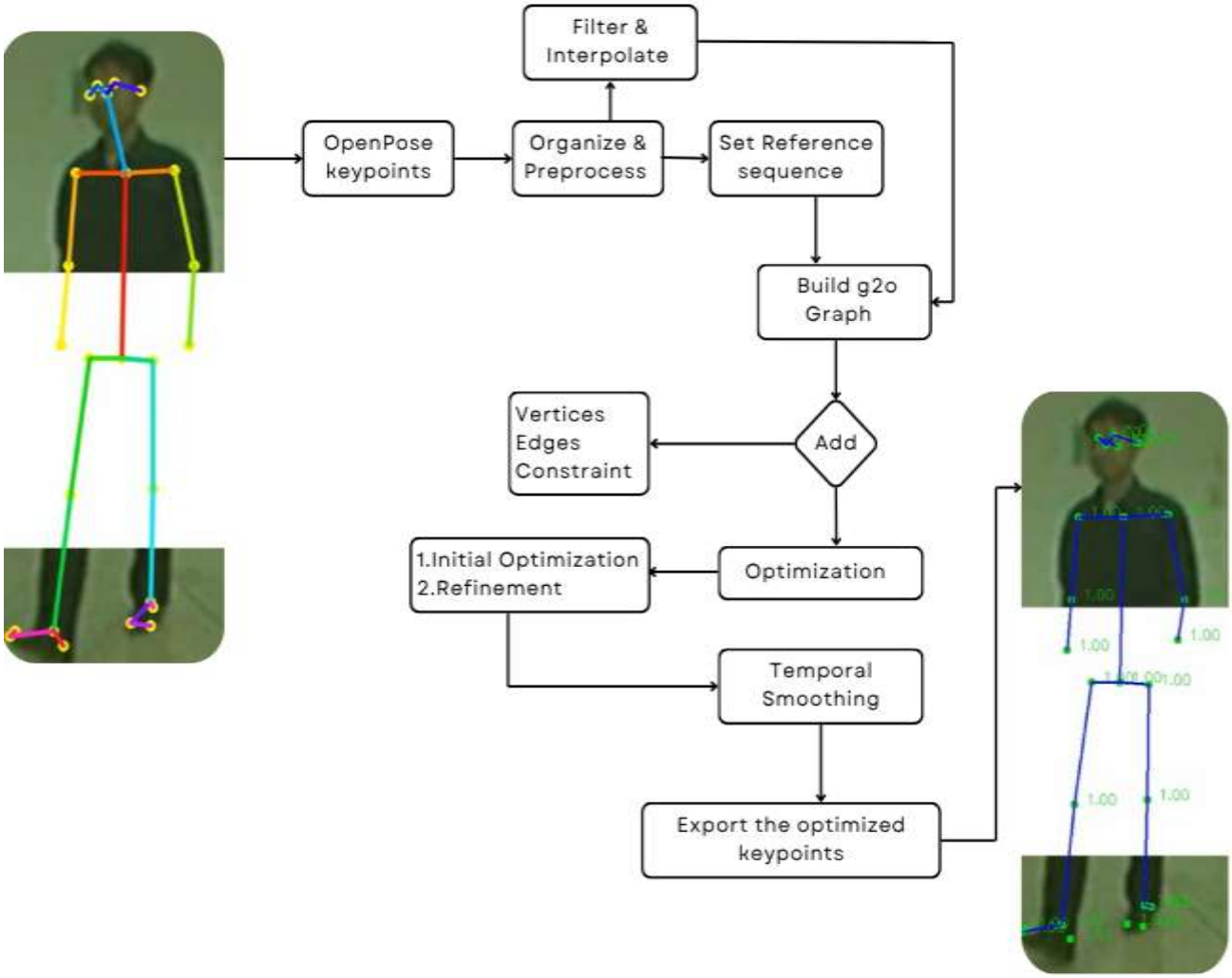

Figure 3.1: AT-G2O Framework Overview