Faculty of Natural, Mathematical and Engineering Sciences
Department of Engineering

Bush House, King's College
London, Strand Campus, 30
Aldwych, London WC2B 4BG
Telephone 020 7848 2145
Fax 020 7848 2851

6CCE3EEP/7CCEMEEP

Individual Project Submission 2024/25

Name: Naghim Ibragimov

Student Number: K22031784

Degree Programme: Electronic Engineering

Project Title: Automated human behaviour in dyadic interactions

Supervisor: Oya Celiktutan

Word count: 13580

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☒ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

RELEASE OF VIDEO

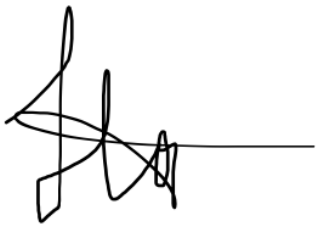
Following the submission of your project demonstration, the Department would like to make it publicly available via youtube. You will retain copyright of the project.

☒ I **agree** to the release of my project video

☐ I **do not** agree to the release of my project video

Signature: (An e-signature is acceptable, i.e., attach your signature as a picture, sign using the draw function)

Date:10/03/2025



"I verify that I am the sole author of this report,
except where explicitly stated to the contrary."

Abstract

This thesis investigates how to improve low-level feature extraction in automated human behaviour analysis in order to improve accuracy and make it easier to extract high-level features. Hand estimate is a significant difficulty in this sector, as it is frequently hampered by obstacles and occlusions. To solve this, a number of low-level feature extraction approaches were tested, including MediaPipe and MMPose. The results show that 2D estimating methods, such those used in MediaPipe, outperform the original approaches. Furthermore, preliminary findings imply that 3D estimation techniques yield even higher accuracy, indicating a promising direction for further research. Improved extraction of low-level hand data is predicted to simplify the mapping to high-level behavioural descriptions, allowing robots to perceive gestures and complicated human interactions.

Contents

List of Figures

Introduction

Nonverbal cues, such hand gestures, are vital for understanding humans in disciplines such as human-robot interaction, remote collaboration, and behavioural research. Automated extraction and interpretation of low-level information, such as hand posture, enables machines to understand and respond to social signals.

This study uses the UDIVA dataset, which contains multimodal recordings of two-person interactions, including synchronised stereo video, audio, and transcripts. Accurate camera calibrations enabled precise stereo triangulation of hand landmarks, resulting in trustworthy 3D data. This dataset allows for the study of natural, real-world behaviours in protracted discussions.

Motivation

SiamRPN++ and other traditional 2D hand tracking methods struggle with occlusions, overlapping hands, and large movements.

Limitations make 2D analysis inaccurate, especially when the hand is partially obscured or outside the frame. Using 3D triangulation with stereo camera views, hand landmarks can be reconstructed even when one camera's view is not functioning. Multi-view fusion improves both spatial accuracy and robustness in many applications.

Improving low-level hand traits helps humanoid robots understand and interpret motions more intuitively. Improved hand tracking allows for more efficient and accurate human-like interactions.

Project Aim

This study aims to improve the extraction of low-level hand features from dyadic interactions using the UDIVA dataset. The project investigates the differences between 2D and 3D hand pose estimation techniques, paying particular attention to the effects of single-view versus multi- view setups. It also explores the strengths and weaknesses of each method through systematic evaluation and analysis.

To improve landmark accuracy and reliability, the project introduces several enhancements, landmark filtering, which help with hand landmarks, confidence-based filtering to remove low-certainty keypoints, and stereo triangulation to reconstruct 3D hand landmarks from dual-camera perspectives. These improvements are benchmarked against the SiamRPN++ baseline, which represents the prior 2D-only hand identification technique utilised in UDIVA-related work.

7

Tools and Implementation

The project was developed in Python with Visual Studio Code as the primary development environment. MediaPipe was chosen for hand land-mark extrac-

tion due to its quick and lightweight deployment. Data manipulation was supported by NumPy and Pandas, while visualisation and plotting were made easier with Matplotlib. Custom triangulation modules were created to convert 2D keypoints to 3D space. Error validation was used in post-processing. Camera parameters were obtained from UDIVA calibration files. Synchronised stereo frames were then processed to extract 2D landmarks and triangulate them into accurate 3D keypoints.

Report Structure

This report is divided into eight major sections, each addressing a specific stage of the project:

- **Part 1 – Introduction:** Describes the project's motivation, goals, and challenges, such as accurately tracking and reconstructing hand movements during dyadic interactions.
- **Part 2 – Literature Review:** Examines prior research on 2D and 3D hand posture estimation, triangulation approaches, multimodal behaviour modelling, and the significance of hand tracking in social interactions. It also identifies gaps in existing methodologies and supports the project's direction.
- **Part 3 – Dataset and Preprocessing:** Discusses the UDIVA v0.5 dataset, selected sessions, stereo camera setup, and calibration files utilised. It also covers the initial data handling, frame extraction, and preparation stages for 2D keypoint extraction.
- **Part 4 – Methodology:** Outlines the pipeline's primary procedures, including 2D hand landmark extraction with MediaPipe, stereo triangulation of keypoints using calibration data, filtering and error-checking, and a matching ID system to identify individual hands across views.
- **Part 5 – Experimental Setup:** Discusses the selection of test sessions (Talk, Ghost, LEGO, and Animal), as well as the evaluation procedure using visual outputs and reprojection error to assess correctness and consistency.
- **Part 6 – Results and Analysis:** Displays pipeline performance across sessions, including successful 3D reconstructions, failure cases, and detailed reprojection error analysis. Graphs compare filtered and unfiltered results.
- **Part 7 – Discussion:** Considers the system's limitations, the improvements implemented to increase hand-person association, and the overall impact of design decisions. It also proposes future enhancements, such as person-level identification and hybrid tracking methods.

- **Part 8 – Conclusion:** Summarises the project's accomplishments, including the creation of a stereo-based 3D hand reconstruction pipeline, and underlines the importance of accurate hand tracking in understanding human interaction. Suggestions for further work are also provided.

Background and Theory

Introduction to Hand Pose in Social Behaviour

Understanding hand pose is a foundational component in modelling human behaviour in dyadic interactions. Low-level features, such as the hand's position, provide critical input for interpreting higher-level behaviours like gestures, attention, and engagement. Therefore, precise hand landmark estimation is essential in this modelling process.

Why Hands Are a Crucial Low-Level Feature

Hand motions are very crucial among all nonverbal signals. Especially in task-oriented environments such as games or 13 cooperative problem-solving, they facilitate communication by signalling direction, expressing emphasis or emotion, and enabling involvement. Hand gestures, for instance, could indicate requests, corrections, or leadership in a situation when two people are building an object together—as in UDIVA's Lego project [6]. Still, gestures are typically quick and delicate, and their meaning depends much on the situation. Understanding high-level social conduct requires good hand posture estimate. Particularly in downstream activities like behavioural predicting or personality inference [7], a badly caught or blocked hand might throw off interpretation. Aiming to give more accurate and full input for these more general behavioural models, this study directly tackles this by improving 3D hand posture extrac-action.

The Overlooked Role of Hands in Behavioural Modelling

Importantly, while face and body modalities have received considerable attention in both dataset design and model development, the hand modality has often been underemphasised. Yet, hands are a vital channel for non-verbal communication. In dyadic settings, hand gestures support turn-taking, emphasis, object manipulation, and emotional signalling. Inaccurate hand tracking therefore poses a serious limitation to the accuracy of higher-level behavioural inference.

Why This Matters

To understand human behaviour in social circumstances, it's important to observe not only what individuals say but also their bodily movements, gestures, and responses during interactions. In face-to-face interactions, nonverbal cues like hand gestures are crucial for effective communication. These cues help convey many social signals such as emphasis, turn-taking, agreement, reluctance, and emotional expression. To effectively assess cues in an automated system, the visual input, such as hand keypoints, must be accurate and reliable. In behavioural modelling, low-level traits form the basis for higher-level judgements. If the base layer, such as hand landmarks, is noisy, inconsistent, or incomplete, further modelling of purpose, emotion, or involvement will be

unreliable. Improving the fidelity and robustness of hand tracking is crucial for developing reliable behavioural analysis systems. This research addresses the demand for precise 3D hand reconstruction through the use of stereo triangulation. Although 2D approaches can be effective, they are limited by viewpoint and susceptible to errors during occlusion or motion. Using numerous perspectives in 3D estimate allows for the reconstruction of spatially coherent hand positions, even if some landmarks are partially or completely concealed in one image. Adding a spatial component enhances understanding of hand structure and movement across time, which is crucial for evaluating social behaviours. 20 This study improves hand feature quality by geometric triangulation, which combines known camera calibrations and complementing 2D observations to provide trustworthy and explainable results. This improves multimodal interaction modelling and advances the objective of making machine perception more socially aware and contextually grounded.

2D vs 3D Hand Pose Estimation

Surveyed Methods in Hand Pose Estimation

Recent assessments on hand pose estimation provide a more thorough description of the fundamental challenges, and model types utilised in the field. Studies have investigated numerous approaches, including RGB-based, depth-based, and hybrid RGB-D methods, as well as the use of CNNs, RNNs, and graph-based architectures for learning-based posture estimation [3], [4], [5]. These evaluations also highlight persistent difficulties like as occlusion, viewpoint fluctuation, and intricate hand- object interactions, all of which have a large impact on the accuracy of both 2D and 3D computations.

RGB-based approaches use regular colour images, while RGB-D methods use both colour images and depth data (these reveal how far the object are).

CNNs (Convolutional Neural Networks) are used to recognise patterns in images, whereas RNNs (Recurrent Neural Networks) are better at recognising sequences, like time-based or spoken data

Hand pose estimation can be done in either 2D or 3D space. In 2D estimation, hand joints are detected in pixel coordinates (x, y) from a single camera view. While this can be effective under controlled conditions, it is highly susceptible to occlusions, self-blocking, or limited field of view. If the hand is partially obstructed or moves out of frame, the extracted features may be incomplete or entirely missing, making it difficult to infer any meaningful gestur. 3D estimation, on the other hand, attempts to estimate the position of each joint in three-dimensional space (x, y, z), frequently employing several camera views. When stereo or multi-view setups are used, each camera offers a different perspective, and missing data from one view can be accounted for by the others. This results in much higher reliability, particularly in real, crowded situations.

Challenges in 3D Estimation

3D estimation presents new issues, including depth uncertainty when converting 2D points to 3D from a single perspective. Wang et al. (2021) found that a single 2D position can result in multiple 3D configurations, making estimation difficult without additional views or temporal context [2]. Improper handling of occlusions and inaccurate detections can lead to poor triangulation when merging information from various views [2].

General Approaches to Hand Pose Estimation

Over the past decade, there has been tremendous advancement in 3D hand posture estimation, including monocular and multi-view techniques. Monocular approaches, which use a single RGB image, are lightweight but have constraints including depth ambiguity, occlusions, and finger similarities, making reliable 3D prediction challenging [8, 9]. To address these issues, multi-view solutions have gained popularity. These algorithms use synchronised camera inputs and triangulation to rebuild 3D joint locations from 2D keypoints. Triangulation is interpretable and modular, but requires correct camera calibration and consistent 2D detections, which can be challenging in real-world interactions [10].

Triangulation Process

Introduction to Triangulation

Triangulation is a fundamental technique in multi-view geometry, used to determine the three-dimensional (3D) position of a point in space by observing it from multiple two-dimensional (2D) projections. This concept is essential in computer vision, it is particularly useful for highend applications such as human pose estimation, object reconstruction, and robotic navigation. When a 3D point is observed from two or more calibrated camera views, each camera records a projection of that point onto its image plane. Triangulation is the process of computing the 3D location of the original point that best corresponds to these multiple 2D projections.

Linear Triangulation: Direct Linear Transform (DLT)

One of the most widely used methods for triangulation is the Direct Linear Transform (DLT), in other workd (OpenCv Triangulation) which estimates the 3D coordinates of a point by solving a set of linear equations derived from the known projection matrices of each camera. Each projection matrix $\mathbf{P}_i \in \mathbb{R}^{3\times4}$ maps a 3D point $\mathbf{X} \in \mathbb{R}^4$ in homogeneous coordinates to its 2D image position $\mathbf{x}_i \in \mathbb{R}^3$:

$$\mathbf{x}_i = \mathbf{P}_i\mathbf{X}$$

To eliminate the unknown scale factor in the projection, the cross product is applied, resulting in:

$$\mathbf{x}_i \times (\mathbf{P}_i \mathbf{X}) = 0$$

This produces two linearly independent equations per camera view. By combining equations from at least two views, a homogeneous linear system $\mathbf{AX} = 0$ is formed. This system is solved using singular value decomposition (SVD), yielding the 3D point $\mathbf{X}$ in homogeneous form, which is then converted into Cartesian coordinates by:

$$\mathbf{X} = \left[\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right]$$

The DLT method is computationally efficient and easy to implement. However, it does not model uncertainty or measurement error and may yield suboptimal results in noisy or imperfectly calibrated conditions [14].

Refinement through Bundle Adjustment

To improve the accuracy of triangulated points, especially in real-world scenarios with noise or slight misalignment, the initial DLT estimate can be refined using a non-linear optimisation method known as bundle adjustment. This technique minimises the reprojection error, defined as the difference between the observed 2D projection of a 3D point and the predicted projection of the estimated 3D point through the camera model.

The total reprojection error E over N views is defined as:

$$E = \sum_{i=1}^{N} \|\mathbf{x}_i - \pi_i(\mathbf{X})\|^2$$

Where $\mathbf{x}_i$ is the observed 2D keypoint, and $\pi_i(\mathbf{X})$ is the projection of the 3D point $\mathbf{X}$ onto camera i's image plane, calculated as:

$$\pi_i(\mathbf{X}) = \frac{\mathbf{P}_i \mathbf{X}}{(\mathbf{P}_i \mathbf{X})_3}$$

This non-linear least squares problem is solved using iterative methods. Bundle adjustment achieves significantly greater accuracy than linear triangulation and is considered a state-of-the-art method for refining 3D structure and camera pose simultaneously [15].

Confidence-Weighted Triangulation

Contemporary methods often incorporate detection confidence into the triangulation process. When each 2D observation $\mathbf{u}_t$ is accompanied by a confidence score c_t, more reliable observations can be weighted more heavily. This leads to improved robustness, particularly in scenarios with partial occlusion or inaccurate detections.

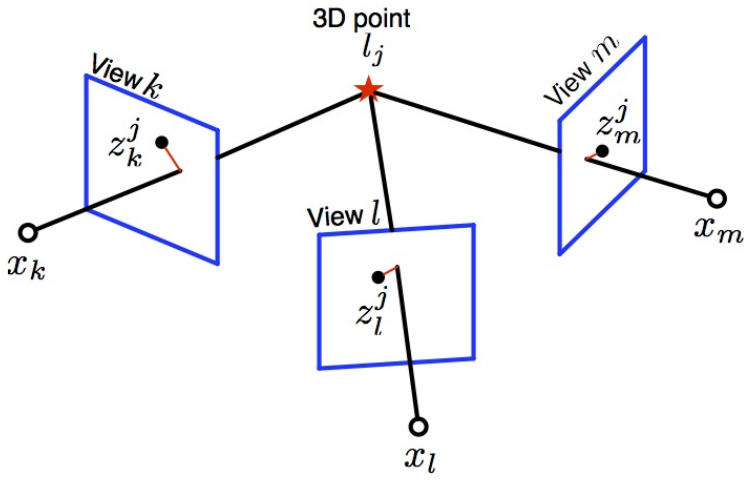


Figure 1: Visualisation of bundle adjustment and Jacobians [**?**].

Liao et al. [17] proposed a formulation where the triangulated point $\mathbf{p}'$ is computed as a function of the 2D keypoints $\mathbf{u}'_t$, their associated confidence scores c_t, and the camera projection matrices Π_t:

$$\mathbf{p}' = \text{Triangulate}\left(\{\mathbf{u}'_t\}_{t=1}^T, \{c_t\}_{t=1}^T, \{\Pi_t\}_{t=1}^T\right)$$

This confidence-aware formulation ensures that low-quality or noisy detections contribute less to the final 3D estimate, while higher-confidence keypoints dominate the reconstruction. This method improves both accuracy and stability across a wide range of conditions [17].

Importance of Camera Calibration

Triangulation also requires on appropriate camera calibration, which supplies the intrinsic parameters (eg rotation and translation across cameras). These parameters 'tell us' how 3D points wish to be projected onto the 2D picture plane of each camera, which makes them an esential aspect of the full 3D modelling process.

Post-Processing and Filtering in 3D Estimation

Even with multi-view triangulation, mistakes can still occur owing to faulty 2D detections, misalignments, or unexpected movements. To address these issues, post-processing techniques are used to stabilise and refine estimated hand trajectories. These filtering approaches play a vital role in boosting the overall reliability of 3D hand pose data. Figure 2 depicts the core concept behind 3D

pose estimation: by combining keypoint detections from several camera perspectives, the system can triangulate the 3D position of each joint. This multi-view technique lowers the impact of occlusion, since missing data in one view can be retrieved from another, and the result is a more full and generalisable representation of the hand in 3D space [2]. Multimodal Interaction

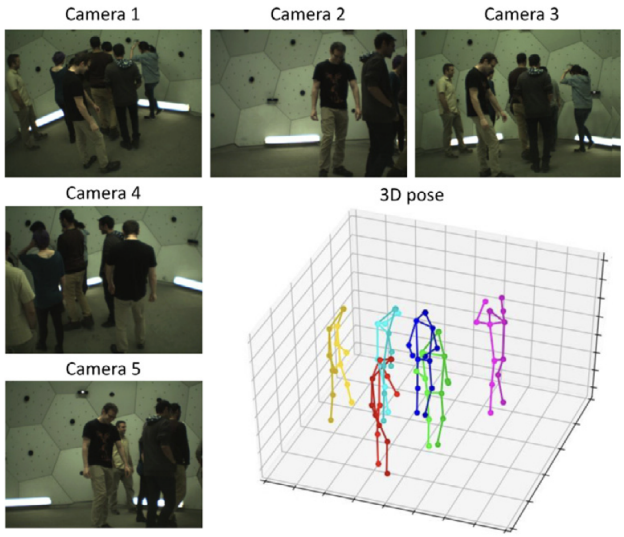


Figure 2: 3D use of methods and tringualtion. Adapted from Wang et al. (2021), used under fair use for educational purposes.

Recent Advances: Learning-Based 3D Hand Pose

Learned Multi-View Models: HandMvNet and MLPHand

Deep learning-based multi-view fusion has lately produced interesting results. Without explicit calibration, HandMvNet is a remarkable model that extracts 2D data from every camera view and combines them using a cross-attention transformer to regress 3D joint and mesh predictions [8].

Leveraging a Multi-View Geometry Feature Fusion (MGFP) block for view-aware fusion, MLPHand presented a more efficient architecture using a Skeleton2Mesh MLP module to create dense 3D hand meshes from sparse keypoints [9].

These models have cutting-edge performance:

- HandMvNet with real-time inference at above 60 FPS obtained MPJPE = 6.98 mm on MVHand.
- On DexYCB-MV, MLPHand achieved MPJPE = 6.00 mm and MPVPE = 6.17 mm with 71 FPS and 90% less parameters than POEM.

The mean Euclidean distance between expected and ground truth joint locations, or MPJPE, (mm).
Equivalent metric for 3D mesh vertices: mean per vertex position error.

Drawbacks and Risks of Learning-Based Methods

Despite their accuracy and speed, these methods come with notable trade-offs:

- They rely on large annotated datasets and are sensitive to domain shifts.
- They require high-end hardware (GPUs) for real-time performance.
- Their black-box nature makes debugging or error correction difficult.
- Errors in 2D keypoint detection can propagate through the entire 3D prediction pipeline.

Even efficient models like MLPHand, although more accessible, still depend on ideal input conditions that are not always met in naturalistic or behavioural settings.

Challenges in 3D Estimation Models

While multi-view 3D reconstruction methods such as HandMvNet and MLPHand show state-of-the-art accuracy on benchmark datasets (e.g., achieving MPJPE values of 6.0–7.0 mm and real-time inference speeds above 60 FPS), they still rely on high-quality input keypoints or feature maps, and often require extensive GPU resources and large annotated training datasets [8, 9]. Their generalisability to unconstrained social environments, such as those captured in UDIVA, is not well established. Moreover, their deep architectures operate as black boxes, limiting transparency and fine-grained control.

Even classical triangulation methods, while more interpretable and generalisable, are only as reliable as their inputs. They require precise camera calibration and suffer if the 2D detections are incomplete or noisy — which is frequently the case in naturalistic, multimodal recordings.

Multimodal Behaviour and Dyadic Interaction

Multimodal Interaction

Human communication is rarely limited to words alone. In dyadic interactions, meaning is conveyed by a combination of verbal and nonverbal cues, including voice, facial expressions, body position, gaze, and gestures. These techniques offer complementary insights into the speaker's emotional state, involvement level, and intents. A person's facial expression or body position may indicate reluctance, even if they vocally agree. Gestures can either strengthen or contradict spoken words. Using only one modality to analyse behaviour, such as audio, can be insufficient and lead to misinterpretation. Combining several data streams,

including video, audio, and text, allows academics to better understand the complexity of human behaviour. Multimodal behavioural analysis is essential in disciplines such as emotional computing and human interactions [6, 7].

Benefits of Multimodal Fusion

Multimodal analysis offers various advantages:

- **Disambiguation:** If a gesture is unclear, clarify its meaning with voice or look.
- **Behaviour grounding** involves aligning gestures with spoken material to convey meaning, such as emphasis or sarcasm.
- **Posture and voice tone** can indicate emotions such as boredom, stress, or engagement, even if words are unchanged.

To fully comprehend behaviour, it's important to consider the context of intent, emotion, and interaction dynamics, not just isolated actions (e.g., raised hand). Improving even one modality, such as visual hand posture, enhances the overall interpretability of human activity [6].

The UDIVA Dataset

The UDIVA Dataset and Multimodal Data

The UDIVA v0.5 dataset (Understanding Dyadic Interaction Via Annotation) is a large, multimodal resource designed to study human behaviour during real-time dyadic interactions. It contains more than 145 recorded sessions involving 134 participants, each engaging in four different conversational tasks:

- *Talk* – open conversation on any topic
- *Lego* – collaborative building task
- *Ghost* – fast-paced visual game
- *Animals* – yes/no guessing game

These tasks were selected to trigger different social behaviours — including cooperation, turn-taking, disagreement, and strategic reasoning — making the dataset rich in both verbal and non-verbal dynamics [6, 7].

Modalities Captured

Each UDIVA session includes synchronised streams of:

- Stereo RGB video (from two frontal cameras),
- High-quality audio using lapel microphones,

- Time-aligned transcripts of speech with speaker identification,
- Automatic 2D facial, hand, and body landmarks,
- 3D gaze estimation,
- Metadata such as age, gender, personality traits, fatigue, and prior relationship between participants.

This multimodal richness enables researchers to study behaviour holistically, using both what is said and how it is expressed across different channels [6].

Relevance to Hand Pose Estimation

Although UDIVA captures multiple modalities, the visual stream, and particularly the hands, play a central role in many of its tasks. For example:

- In *Lego*, participants use hand gestures to point, position, and explain.
- In *Animals*, they may use hand signals to indicate guesses or reactions.
- Even in free conversation (*Talk*), hands support emotional expression and engagement.

Figure 3: Figure 2: Stereo task views from UDIVA [7].

Figure 3: Example frames from the UDIVA v0.5 dataset showing stereo camera views (FC1 and FC2) across four interaction tasks — Talk, Lego, Ghost, and Animals. Each task elicits different types of social behaviours such as conversation, collaboration, competition, and guessing, captured simultaneously from two synchronised camera angles.

Hand Tracking in the UDIVA Dataset

The FrankMocap model handled hand pose estimation in UDIVA v0.5. Missing frames were corrected by filtering and linear interpolation. Hand landmarks were found to be less reliable than face or body landmarks (99.3% and 97%,

respectively), with a final validation accuracy of 80–90% [11]. In fast-paced challenges like *Ghost* or *Animals*, participants experienced left/right hand confusion, missing keypoints, and frame inconsistencies. Although stereo recordings were available, neither triangulation-based or geometry-informed reconstruction approaches were used to enhance the hand data. Corrections used temporal tracking (SiamRPN++) and confidence filtering, which were effective but limited in scope.

UDIVA hand extraction limitations

Hand posture estimation remains challenging, especially when moving from controlled benchmarks to real-world interaction datasets. Low-level visual inputs, such as hand landmarks, are crucial in dyadic social behaviour modelling yet remain one of the most unreliable elements in the pipeline.

Current techniques often start with 2D keypoint extraction from a single RGB view, employing models like MediaPipe Hands or FrankMocap. Although rapid and accessible, these approaches have significant limitations. They are susceptible to occlusion, motion blur, and partial visibility, which are typical in social interactions. Even after post-processing, UDIVA datasets showed inferior accuracy (80-90) percent for hand landmarks compared to facial (99.3) percent and body features (97) percent [11]. This performance gap is especially noticeable in high-movement tasks like Ghost or Animals, where frequent occlusion and quick hand gestures cause significant landmark loss.

Temporal Inconsistency Across Frames

2D keypoints often lack temporal coherence across frames. Even with filters such as One-Euro or SiamRPN++, landmark jitter, disappearance, and identity-switching remain common problems. This disrupts the continuity needed for downstream tasks like gesture recognition or behavioural forecasting. Without reliable frame-to-frame consistency, it becomes difficult to interpret gestures or maintain tracking of specific individuals' hands over time.

A Targeted and Modular Response to These Gaps

In contrast to the limitations found in existing pipelines, this project deliberately adopts a minimal yet highly controlled setup. It uses two stationary cameras with full intrinsic and extrinsic calibration, removing ambiguity in the triangulation process. This foundation ensures geometrically accurate 3D reconstruction of hand landmarks, even in the presence of occlusions.

Additionally, a lightweight post-processing stage integrates filtering techniques such as confidence-based smoothing to enhance stability and reduce noise across frames. To maintain consistency across multiple individuals and hands, the pipeline incorporates a basic hand identity management system, which assigns consistent IDs to each hand across views and time.

Furthermore, while MediaPipe Hands serves as the core extractor, alternative models such as MediaPipe Holistic were also evaluated during early experimentation. This modular design ensures both robustness and transparency, allowing for systematic improvement of low-level hand features within realistic dyadic interaction environments.

Report

Methodology and Implementation

The core objective of this project was to construct an accurate and interpretable pipeline for extracting 3D hand keypoints from stereo videos of dyadic interactions, using the UDIVA dataset. The full process consists of five key stages: data acquisition, 2D keypoint extraction, cross-view hand matching, triangulation (bundle adjustment), and visual validation. Each step was carefully designed to maximise spatial accuracy, consistency, and modularity for behavioural analysis.

1. Stereo Input and Camera Calibration
The process began with the selection of synchronised stereo videos recorded using two fixed frontal cameras. These cameras were fully calibrated, with known intrinsic parameters. This calibration setup enabled geometrically valid triangulation of points across the stereo pair.

2. 2D Hand Landmark Extraction
Hand keypoints were extracted from both camera views using the MediaPipe Hands framework. This model detects up to four hands per frame and estimates 21 hand landmarks for each, alongside handedness classification (left/right) and confidence scores for each keypoint. These detections form the low-level visual foundation upon which the rest of the pipeline builds.

3. Cross-View Hand Matching
To enable triangulation, it is essential to ensure that each detected hand in one view corresponds to the correct hand in the other view. This step involved associating hand detections across cameras using spatial proximity between wrist keypoints, assuming that the stereo setup would preserve relative positions. Only complete detections (i.e. hands with all 21 landmarks) were considered valid for matching.

4. 3D Triangulation and Refinement
Once matched, corresponding 2D landmarks were undistorted using the calibration data, then triangulated using a Direct Linear Transform (DLT) method. The initial 3D estimates were further refined via bundle adjustment, minimising reprojection error using nonlinear optimisation. This stage produced accurate, frame-wise 3D landmark positions in real-world coordinates. Filtering based on keypoint confidence was evaluated.

Video Selection Strategy (pre-fusion)

As a first step, exploratory analysis was performed by analysing multiple sessions from the UDIVA dataset to determine which recordings had the most significant hand activity. The goal was to choose videos that would both portray normal interaction and pose obstacles to the hand extraction pipeline, particularly those

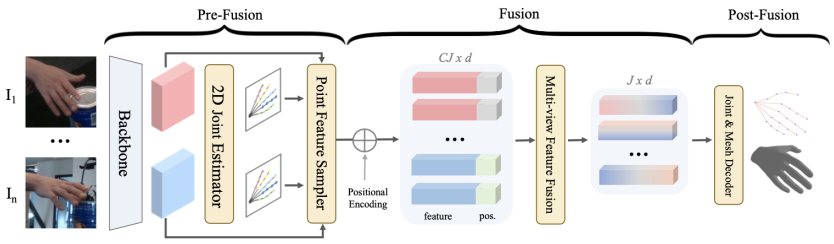


Figure 4: Modular pipeline adapted from HandMvNet [12].

with fast-paced hand movements, occlusions, or frequent exchanges between players.

Recordings of "Talking" activities exhibited minimal hand motion and were relatively simple to process. In contrast, sessions like "Lego" and "Ghost" included highly dynamic scenes involving continual hand movement, object handling, and hand overlap among participants. While these movies were important for understanding the limitations of existing extraction approaches, they were not chosen for final experimentation because the project prioritised validating the reconstruction pipeline above benchmarking across harsh situations. Instead, relatively difficult recordings were chosen to allow for a reliable assessment of stereo triangulation under realistic conditions.

Calibration Usage

To enable accurate triangulation, this project made use of the camera calibration files provided within the UDIVA dataset.

The two primary cameras used in this project FC1 and FC2 are labelled as Z1 and Z2 in the calibration structure. According to the dataset documentation, cameras were calibrated once per day. Although they remained in fixed positions throughout recording sessions, slight shifts could not be entirely ruled out. Each calibration folder is organised by recording date, allowing it to be matched with the corresponding video sessions using the timestamp metadata. These parameters were crucial for undistorting the 2D keypoints and ensuring geometrically consistent 3D triangulation across frames.

Method Selection for 2D Keypoint Extraction (pre-fusion)

Choosing an appropriate method for 2D hand landmark extraction was a critical step in this project, as it formed the foundation for all subsequent processing, including hand matching and 3D triangulation. Several frameworks were evaluated based on accuracy, ease of use, and computational feasibility, especially considering the constraints of working on a MacBook M2 with no dedicated GPU.

Initial Exploration and Framework Comparison
The first candidate explored was OpenPose, a well-established tool capable of detecting multiple people and estimating full-body, face, and hand landmarks. While OpenPose is known for its reliable results and comprehensive output, it quickly proved unsuitable due to its heavy GPU requirements and outdated dependencies. Running it on video sequences without high-end hardware led to significant performance issues and impractical processing times.

MMPose, developed by OpenMMLab, was also tested due to its support for advanced multi-person pose tracking and flexible architecture. It performed well in terms of landmark accuracy and supported both 2D and 3D pose estimation. However, similar to OpenPose, it demanded significant computing power and struggled to run efficiently in a CPU-only environment. In addition, its configuration and model loading processes were more complex, slowing down the development pipeline.

Final Choice: MediaPipe Hands
After testing multiple alternatives, MediaPipe Hands was selected as the final 2D hand tracking method. Developed by Google Research, it is a lightweight and highly efficient solution that operates in real-time on both CPUs and GPUs. It predicts 21 hand landmarks per hand, along with handedness (left/right) classification and confidence scores for each keypoint. Unlike the heavier frameworks, it required no training or complex setup and was fully accessible using Python on the MacBook M2.

MediaPipe Hands stood out due to its:

- **Efficiency**: Runs in real time even on low-spec machines.
- **Ease of integration**: Pre-trained and deployable with minimal setup.
- **Clarity of output**: Provides landmark coordinates, confidence levels, and hand classification.

However, it also came with limitations. It does not support persistent hand IDs or person association, which means that in multi-person scenes, it is not possible to automatically determine which hand belongs to which person. This limitation was particularly relevant for dyadic interactions like those in the UDIVA dataset. Additionally, performance degrades in cases of occlusion or extreme angles.

To address these issues, the project later implemented custom logic for cross-view hand matching and ID assignment. Despite its constraints, MediaPipe Hands offered the best trade-off between performance, accessibility, and functionality, and served as a strong foundation for the stereo 3D reconstruction pipeline [13].

Extracted Keypoint Data and Structure(pre-fusion)

As part of the 2D landmark extraction process, each frame in the stereo video yielded a set of hand keypoints, represented in a structured format. The data

extracted for each visible hand included the following key fields:

- **frame_index**: Indicates the frame number in the video sequence, used for temporal alignment across views.
- **hand_type**: Specifies whether the detected hand is classified as "Left" or "Right," as predicted by MediaPipe Hands.
- **landmark_index**: An integer between 0 and 20, identifying each of the 21 standard hand landmarks (e.g., wrist, fingertips, joints).
- **x, y**: The 2D pixel coordinates of the landmark in the frame, used as input for stereo triangulation.
- **landmark_confidence**: A visibility or confidence score ranging from 0 to 1, where higher values indicate greater model certainty in the detection.

These fields were extracted per hand, per frame, across both camera views. They served as the core input for later stages of the pipeline including hand matching and 3D reconstruction.

Throughout development, multiple variations of this dataset structure were tested. Several iterations included additional body-related keypoints, such as shoulders or head, particularly when experimenting with MediaPipe Holistic for possible hand-to-body assignment. However, these approaches often introduced inconsistencies and increased complexity, especially when occlusions occurred or body pose estimation was unreliable.

After extensive trial and error, the most robust and generalisable version of the data included an additional field for a matched id a custom identifier used to link corresponding hands across both stereo views. This proved far more effective than relying on other low-level body landmarks. Matching hands using this ID system allowed the triangulation process to associate correct keypoints between camera views without needing full-body pose alignment, and reduced ambiguity in multi-hand scenes.

This structured, per-frame hand keypoint data particularly with the addition of the matched ID the backbone for reliable stereo triangulation and consistent tracking across frames.

Very first extracted data where: Hand index identifies the hand, (ie left or right or 0,1) and landmark index are the keypoints indexes for hand its 0-21. This was only computed for maximum of two hands.

Triangulation Process (fusion)

Once corresponding 2D hand landmarks were identified across both camera views, the next step was to compute their 3D positions in space using stereo triangulation. This required choosing a triangulation strategy that would deliver high spatial accuracy while remaining compatible with the noise and occasional imprecision of keypoint detections.

frame_index	hand_index	landmark_index	x	y
0	0	0	668.2888793945310	399.9293804168700
0	0	1	707.3831176757810	414.6039390563970
0	0	2	733.9820861816410	411.9079542160030
0	0	3	753.3047485351560	414.17036533355700
0	0	4	768.7144470214840	421.6109848022460
0	0	5	739.2962646484380	363.3918571472170
0	0	6	772.9296875	384.8030090332030
0	0	7	781.5099334716800	405.9497594833370
0	0	8	785.3880310058590	422.8501224517820
0	0	9	735.5399322509770	362.6625967025760
0	0	10	776.2063598632810	385.6105041503910
0	0	11	784.9668884277340	409.605975151062
0	0	12	785.6172180175780	428.5772180557250
0	0	13	730.8348846435550	369.6108913421630

Figure 5: Figure 4: Example extraction output.

Triangulation Method Selection

Two main triangulation approaches were considered during development:

- OpenCV's Direct Linear Transform (DLT) triangulation via `cv2.triangulatePoints`
- Bundle Adjustment using nonlinear optimisation via `scipy.optimize.least_squares`

The DLT approach uses camera projection matrices to estimate 3D point positions from 2D points, making it quick and easy to implement. However, it requires accurate input keypoints and does not account for reprojection error or confidence ratings. This might be problematic when working with real-world data with occlusions or landmark jitter. Bundle adjustment was the primary triangulation approach used to increase accuracy. This technique improves 3D point estimates by reducing reprojection error, which is the difference between the original 2D keypoints and the estimated 3D point projected onto the image plane. The optimisation process iteratively adjusts each landmark's 3D position to coincide with both camera perspectives, considering distortions, camera intrinsics, and projection geometry. This resulted in more accurate and consistent 3D reconstructions, particularly in frames with faulty 2D detections.

Use of Camera Calibration (Z1 and Z2)

To perform accurate triangulation, full camera calibration was required. The UDIVA dataset provided this information in the form of `.yaml` files for each

camera — labelled Z1 (for FC1) and Z2 (for FC2). These files included:

- **Intrinsic matrices** (K1 and K2), describing internal camera parameters like focal length and principal point
- **Distortion coefficients** (D1 and D2), used to undistort the raw 2D keypoints
- **Extrinsic parameters**, in the form of a rotation matrix (R) and translation vector (T), describing the spatial relationship between the two cameras

All 2D keypoints were first undistorted using the provided intrinsics and distortion coefficients. These corrected coordinates were then used in both DLT and bundle adjustment pipelines. The extrinsic parameters (R and T) were critical in defining the stereo geometry, allowing accurate triangulation of each landmark from the two views. This process produced the 3D hand keypoints for each frame in real-world coordinates, forming the basis for subsequent projection and behavioural analysis.

Post-Processing and Filtering (post-fusion)

To improve the accuracy and reliability of 3D hand reconstruction, especially in the presence of occlusion or uncertain detections, two main post-processing strategies were applied: landmark validation and confidence-based filtering.

Landmark Consistency Checks and Occlusion Handling

To ensure only complete and geometrically plausible hand detections were used for triangulation, each detected hand was evaluated based on the presence of all 21 expected hand landmarks as returned by the MediaPipe Hands model.

- A hand detection was only considered valid if it contained all 21 landmarks, including critical joints such as the wrist (index 0) and fingertips (indices 4, 8, 12, 16, 20).
- Hands with missing landmarks or visibly incorrect positioning (e.g., broken structure due to occlusion or fast motion) were excluded entirely from the triangulation process.
- This validation was particularly useful for dealing with challenging frames where:
  - hands crossed over each other,
  - hands were partially visible due to the field of view,
  - or participants were gesturing rapidly.

By discarding incomplete or structurally invalid detections, the pipeline avoided introducing faulty geometry into the 3D reconstruction.

Confidence-Based Landmark Filtering

In addition to checking for structural completeness, each landmark included a confidence score provided by MediaPipe. This score reflects how confident the model is in the location of each keypoint.

A manual confidence threshold was applied to improve the quality of landmarks used for triangulation:

- Landmarks with confidence below 0.6 were considered unreliable and ignored.
- Entire hands were excluded if the average confidence was too low, or if key points such as the wrist and index fingertip did not meet the threshold.
- This helped eliminate noisy detections due to motion blur, poor lighting, or incorrect model predictions.

Hand ID Assignment and Person Association (pre-fusion)

Even though the logic of assigning hands to individuals chronologically fits just after 2D keypoint extraction and before triangulation, this section is included later due to its central importance and the extensive experimentation it required throughout the development of the pipeline.

Initial Assumptions and Early Challenges

At the beginning of the project, the goal was straightforward: apply the camera calibration parameters to triangulate MediaPipe's 2D hand keypoints into 3D space. Early testing focused only on this technical correctness, without logic to assign hands to individuals.

However, early reconstructions showed many errors — hands floating in wrong positions or incorrectly aligned — even though calibration data (Z1–Z2) seemed correct. This led to a deep inspection of the calibration files and tests with different matrix combinations, but the issue remained.

Deeper Inspection into the Extraction Phase

The cause was ultimately traced back to MediaPipe Hands, which provides no person ID — only whether a hand is left or right. When Person A's left hand appears in one view and Person B's left in the other, the system wrongly matches them. This produces completely invalid triangulated outputs, even if all detections were spatially accurate.

Post-Processing Attempts and Filtering Improvements

To manage this, post-processing was applied. Low-confidence keypoints were removed based on MediaPipe's landmark-level scores. Additionally, partial hands (with missing keypoints) were filtered out.

Attempt 1: MediaPipe Holistic – Shoulder Matching Logic

An early solution involved using MediaPipe Holistic to get shoulder landmarks. Midpoints between left and right shoulders were used to infer person location, and hands were then assigned to the nearest body. However, this failed in many cases due to missing shoulder detections, off-camera persons, or occlusions. Only one person was often detected, making the logic unreliable.

Figure 6: Shoulder keypoint extraction (MediaPipe Holistic (Attempt 1)).

Attempt 2: Assumption-Based Matching

This approach assumed two people were always visible:

- Shoulder clusters were sorted by X-coordinate to assign persons.
- Hands were matched by proximity to shoulder clusters.

This worked in ideal conditions but failed with partial detections or perspective distortion — especially during fast movements or occlusion.

Performance and Hardware Limitations

Running Holistic on a MacBook M2 (CPU-only) was computationally heavy. Splitting hand and shoulder detection helped but made matching more complex. High-motion scenes like LEGO tasks still caused invalid matches.

frame_index	matched_id	landmark_index	X	Y	Z
0	0	0	30.893103603469800	61.65994926791510	1488.1602465004300
0	0	1	64.72482266620060	78.01840269191660	1496.6352647437100
0	0	2	85.87333330320660	85.31618326624820	1489.0941534190800
0	0	3	100.05531429474500	95.03760346045440	1472.9347596170600
0	0	4	111.13659858266100	107.02599237606000	1460.027989385200
0	0	5	86.20287700111530	55.137354218467000	1446.7139388931400
0	0	6	112.32680031403300	81.57960395943720	1439.4196271290300
0	0	7	119.47803426817900	101.39686972251800	1445.477346440640
0	0	8	123.31808589061400	116.30771359113100	1454.1819513268900
0	0	9	82.24732160228920	54.01391612911880	1433.3803028186300
0	0	10	112.43530271588800	82.46661763070250	1410.5526386884900
0	0	11	120.88987768778200	100.3688173713040	1421.9332011463100
0	0	12	123.5164421077350	112.23122368537800	1436.8653373701900
0	0	13	78.08026138522940	57.86844348317280	1426.6270056468300
0	0	14	107.94448400340900	84.00257124330730	1404.5252992043200
0	0	15	117.60680596829000	100.00536851356300	1415.6066867632500
0	0	16	120.69678681434500	109.49135871643400	1429.2502001587200
0	0	17	75.08658046726070	64.44928152782030	1425.3537171882000

Figure 7: Matched ID example (Attempt 2)

Final Approach: Matched ID + Filtering

The final, most stable method:

- Assigned consistent matched IDs based on spatial position and tracking.
- Used confidence-based filtering to reject low-certainty landmarks.

This approach delivered the best results, particularly in moderate-movement scenarios like talking sessions.

Reflections and Future Considerations

Given more time, combining OpenPose (for person ID) and MediaPipe (for hand landmarks) would be optimal. OpenPose provides global consistency, and MediaPipe provides detailed hand data. The key takeaway: extraction tools should be selected not just for accuracy, but for how well they match the project's constraints and goals.

Limitations in Full Reconstruction

Although extraction, matching, and filtering techniques have improved, there are still problems that impact the quality and completeness of 3D reconstruction. One issue was the prevalence of severe outliers. In several frames, the triangulation method resulted in 3D points that were not within the expected range, such as floating far away from the hand or outside the picture. These were not common, but when they happened, they distorted the picture and emphasised mismatches in stereo keypoints. Occlusion and limited visibility were more common concerns. During sessions like the LEGO assignment, participants frequently interacted with objects that covered their vision of one or both hands in at least one camera. In such circumstances, MediaPipe failed to extract useful landmarks and the related 3D model.

The most tough obstacle comes from rapid movement and severe hand positions. Rapid motions or awkward angles made it impossible for MediaPipe to identify consistent 2D landmarks in either view. Even if both cameras generated outputs, they could be misaligned or partially incorrect, resulting in unsatisfactory triangulation findings.

There were also some small inconsistencies in 3D geometry. While many hand reconstructions appeared to be correct, others had abnormal finger spacing or proportions. This could be attributed to depth ambiguity or minor problems in stereo calibration. The calibration files for the UDIVA dataset were prepared using static setups (as stated in the README), and while this setup was generally reliable, this project assumed zero per-frame rotation and translation during reconstruction. Incorporating the full R0/T0 matrices from the calibration files may enhance accuracy, but doing so over thousands of frames would necessitate extensive data handling and frame-by-frame modification.

Despite constraints, the pipeline provided relevant results in many frames, especially under controlled conditions like *talking* sessions. Future enhancements could include incorporating per-frame extrinsic corrections, adding temporal smoothing across frames, or utilising hybrid models that mix geometry and deep learning priors to produce more robust 3D output.

Technical Setup Summary

The implementation was carried out entirely in Python, using a combination of open-source libraries and custom scripts to process the UDIVA recordings and reconstruct 3D hand keypoints. The following tools formed the core of the pipeline.

The core Python libraries used were:

- **MediaPipe:** used to extract 2D hand landmarks and handedness.
- **OpenCV:** handled stereo triangulation, calibration loading, and visualisation.
- **NumPy and SciPy:** supported array operations and matrix handling.
- **Matplotlib:** enabled plotting and analysis of reprojection error.
- **Pandas:** used for reading and writing landmark data in CSV format.

Custom scripts developed as part of the pipeline included:

- `extraction.py` – extracts 2D hand keypoints and handedness using MediaPipe.
- `matchID.py` – assigns consistent matched IDs to detected hands across both views.
- `triangulation.py` – performs triangulation using the calibration parameters.
- `visualisation.py` – projects and displays 3D points over 2D frames.
- `combination.py` – combines and aligns detection results from both cameras.
- `error.py` – calculates reprojection error and performs filtering analysis.

These scripts were not standalone tools but part of a tightly linked structure. For example, triangulation could not proceed without first running `extraction.py` and `matchID.py`, as both 2D keypoints and consistent hand identifiers were required to ensure correct triangulation.

The UDIVA dataset also provided pre-recorded video libraries and accompanying camera calibration files (`.yaml`), which were used throughout the process. Calibration data from cameras Z1 and Z2 was used to compute the stereo geometry and reconstruct 3D coordinates.

This structure ensured that each component had a clearly defined role within the broader pipeline, while still being modular enough to allow testing on multiple sessions and conditions.

Results and Evaluation

Starting the 3D reconstruction process using MediaPipe Hands yielded good results, as the pipeline worked as predicted at all important steps, from 2D keypoint extraction and triangulation to visualisation and reprojection error evaluation. The system performed exceptionally well in instances with minimal hand movement, not because the approach struggled with motion in general, but rather because hand discrimination was difficult when numerous persons were present.

Figure 8: 3D triangulation success in stable environments. Red is tringualted, Green is original 2D extraction

One of the main issues was MediaPipe's inability to tell which hand belonged to which individual, especially during close contact. This was clear in the LEGO task recordings, as people regularly delivered LEGO pieces to each other, causing their hands to overlap or come into close contact. These interactions resulted in inaccurate triangulations because the algorithm struggled to appropriately identify each hand with the appropriate participant across stereo viewpoints.

Figure 9: Failed identification, Triangulated wrong hands. s. Red is tringualted, Green is original 2D extraction

Another issue discovered during testing was frames in which a hand was seen in one camera but not the other. In such circumstances, the computer would still try to triangulate the visible hand by comparing it to an unrelated or inaccurate detection in the opposite view. Although the use of matching IDs improved this by ensuring some consistency in hand assignment, it was not failsafe. There

were still frames where incorrect matches passed through, resulting in invalid 3D outputs.

Figure 10: triangulation of invisible hand. s. Red is tringualted, Green is original 2D extraction

However, when the algorithm correctly identified both the hand and the matching individual, triangulation was extremely successful. These examples clearly indicated that stereo calibration and the assumption of fixed camera extrinsics (no per-frame rotation or translation) were valid under the test conditions. One of the most consistent and successful sessions was the talking video, which kept hands near to the body and in a reasonably static frame. In this case, the majority of 3D points were correctly reconstructed, demonstrating the pipeline's capacity while landmark visibility and identification were stable.

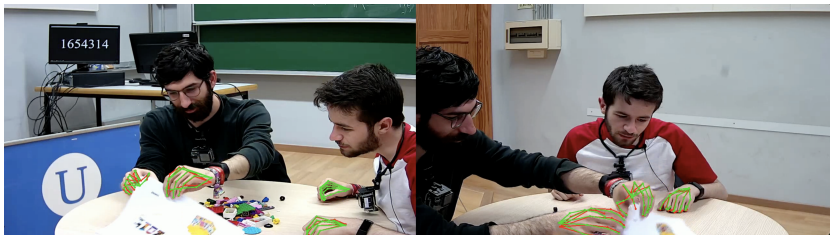


Figure 11: Successful triangulation in obstructed views. Red is tringualted, Green is original 2D extraction .

Reprojection Error Visualisation and Evaluation of Data

Reprojection error was computed for each frame in order to conduct a more in-depth analysis of the quality of the triangulated three-dimensional points. It is the pixel difference between the original two-dimensional landmark and the three-dimensional point that is reprojected back onto each camera plane that is measured by this inaccuracy. Talk, Ghost, LEGO, and Animal were the four different types of graphs that were created for the various UDIVA sessions. These charts illustrate the error trends for Cam1 and Cam2 over a period of one thousand frames, illuminating the ways in which performance varies depending on characteristics such as motion, visibility, and interaction type.

Talk Session: This was the session that was the most consistent overall. The majority of frames maintained modest reprojection errors that were less than 25 pixels, as can be seen in the first plot displayed. Approximately between frames 80 and 150, a significant spike is observed, which is most likely the result of a hand being mismatched or obstructed, resulting in a significant misalignment. The results were extremely constant, with the exception of this spike and one other little peak that occurred close to frame 820. The fact that the pipeline functions effectively when the hands are visible, close to the body, and remain reasonably steady is demonstrated by this demonstration.

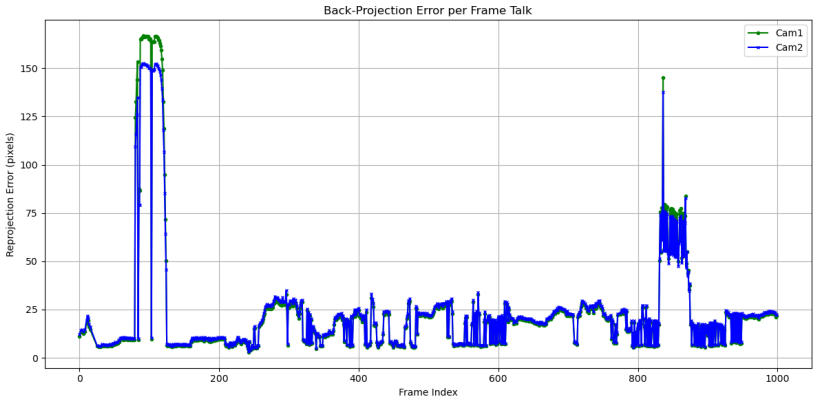


Figure 12: Reprojection error visualisation for the Talk session (Cam1 and Cam2).

Ghost Session: The Ghost session was characterised by a greater amount of motion and variety, which resulted in substantially higher and more frequent spikes in the amount of reprojection error. Several frames had values that were greater than fifty pixels, and there were multiple strong peaks that occurred between frames 200 and 900. These can be attributed to rapid motions, hands going in and out of view, and the possibility of misassignments occurring as a result of gestures that overlap or data that is absent from one camera. In spite of these difficulties, baseline errors in frames with a lower level of dis-

traction remained acceptable, demonstrating that certain frames still produced satisfactory triangulation.

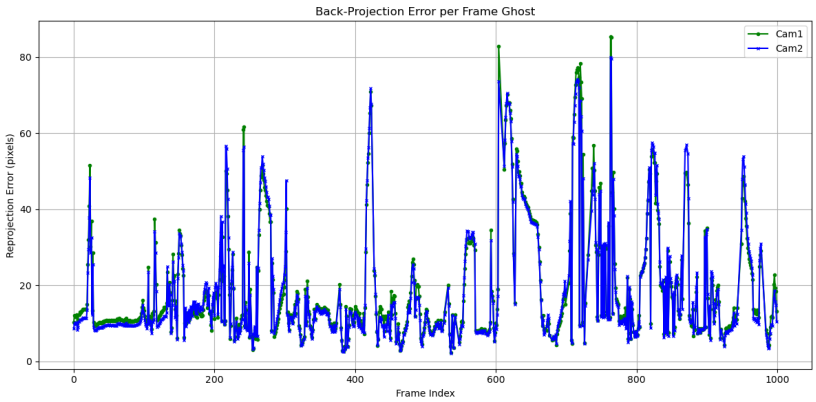


Figure 13: Reprojection error visualisation for the Ghost session (Cam1 and Cam2).

LEGO Session: The LEGO Session was, as you may have anticipated, the most difficult of all the sessions. Throughout the entirety of the time period, the graph displays a dense series of highly significant error spikes. The characteristics of the task, which included frequent handovers, overlapping hands, and high motion, resulted in a significant number of cases in which hands were either incorrectly detected or obscured in a single view. As a result, there were regular mismatches, particularly in situations where both hands of two individuals were interacting in close proximity to one another. Triangulation remained unstable in the majority of frames, despite the fact that matched IDs helped eliminate part of the inaccuracy.

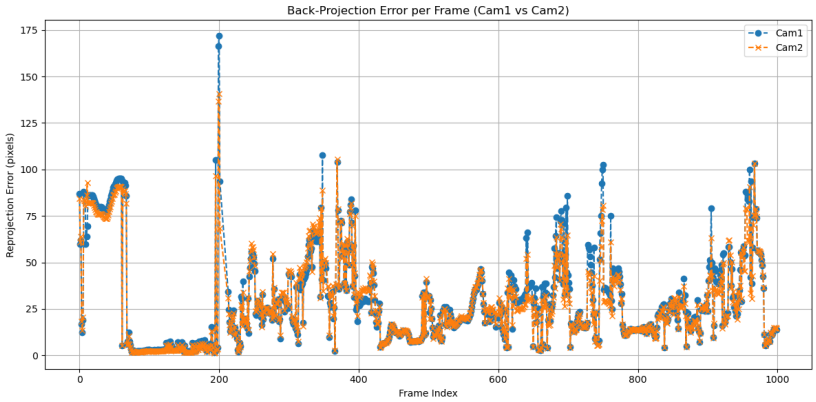


Figure 14: Reprojection error visualisation for the LEGO session (Cam1 and Cam2).

Animal Session: During the Animal Session, there were moderate amounts of errors encountered. There were spikes, but they were not as regular or as severe as the ones that caused the LEGO or Ghost sessions. Errors, with the exception of sharp outliers, were seldom more than thirty pixels in size. These errors were frequently brought on by incomplete hand visibility or frames in which MediaPipe failed to capture certain landmarks. With findings that were closer to those of the Talk session, this session did better than Ghost and LEGO in terms of overall performance. The results indicated that there was modest movement and clearer sight.

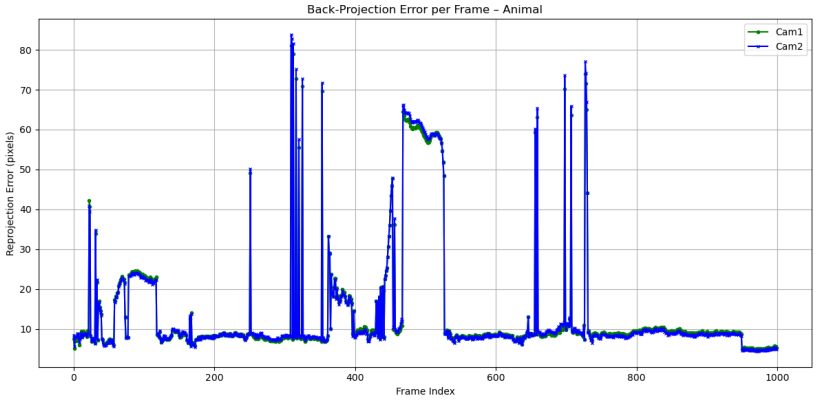


Figure 15: Reprojection error visualisation for the Animal session (Cam1 and Cam2).

Conclusion

This project aimed to increase the reliability of hand posture estimation in dyadic interactions by utilising stereo triangulation of 2D landmarks retrieved with MediaPipe. Using calibrated camera setups and a modular Python-based pipeline, the system was able to rebuild 3D hand keypoints with reasonable accuracy, even in scenarios with two interacting humans. The pipeline included matching ID tracking, confidence-based filtering, and structured reprojection error analysis, all of which helped to produce more consistent results in sessions when hands stayed visible and relatively immobile.

Despite ongoing issues, particularly in sessions with fast movement, occlusion, or overlapping gestures, the pipeline consistently generated meaningful reconstructions across a wide range of frames. Triangulation accuracy varied depending on the session, with the highest results reported during slower-paced interactions (for example, the chatting session) and more frequent mismatches during high-interaction tasks such as LEGO or Ghost.

One of this work's most significant contributions is its emphasis on the often neglected but crucial role of hands in social behaviour modelling. By addressing low-level anomalies in hand posture estimate, this effort enables more robust behavioural annotation and future machine perception systems capable of interpreting human interaction in a deeper and more contextually aware manner.

Looking ahead, the system could be improved by using temporal tracking to ensure consistency over time, or by combining classical geometry with learning-based models for more adaptive hand matching. Incorporating OpenPose for person-level identification and testing with depth sensors are both intriguing directions.

This project has also demonstrated how important technique selection is in the early stages of design. Choosing the correct extraction framework was ultimately more crucial than imagined, influencing every stage of the process. While the hand-matching problem is only partially solved, our work provides the framework for future enhancements, which might include a specific identification algorithm for MediaPipe Hands. It is hoped that others will build on these findings, adapt the technique, and strive for systems that can perceive and understand complicated human interactions.

Professionalism and Responsibility

This project was carried out with a clear dedication to professional integrity, data security, and ethical responsibility. Working with human behavioural data, especially from a multimodal dataset like UDIVA, raised a number of ethical concerns about privacy, permission, and data security.

Before accessing and working with the UDIVA v0.5 dataset, a legal agreement was reached with its producers. This agreement clearly defined the terms of use, including limitations on the external sharing, publication, or transfer of any visual or identifiable data involving participants. Only photographs and

examples from persons who had explicitly consented to limited academic usage (such as inclusion in reports or dissertations) were used in this project, and only within the parameters set under the agreement. No personal information, identity-related metadata, or facial photographs were published or distributed outside of this regulated academic setting.

All visualisations and outputs created during this project were handled in a secure and responsible manner. Where facial data or potentially identifiable traits were included, those frames were either removed or anonymised in accordance with the dataset usage requirements. Any reproduced visuals, such as stereo views or hand visualisations, were utilised solely for technical demonstration within the scope of this study and were never intended for public distribution or internet sharing.

From a technical and scientific basis, the project followed sound engineering practices. All code and procedures were written with clarity, transparency, and modularity in mind, so that the system may be examined, updated, or reused by others in a responsible manner. Open-source libraries like MediaPipe and OpenCV were used using acceptable licenses, and all external tools, models, or datasets were cited.

Cybersecurity was also considered during implementation. The UDIVA dataset and all outputs were stored locally on encrypted storage, with no cloud uploads or third-party hosting involved at any time. This guaranteed that sensitive data was contained within a secure and university-compliant environment.

Finally, this work aims to make a good and responsible contribution to the larger field of human-computer interaction and behavioural modelling. By handling sensitive data with care and honouring participant rights, the project not only met academic requirements but also met the standards expected of professional engineering and research activity.

Bibliography

References

[1] Liao, Z., Zhu, J., Wang, C., Hu, H. and Waslander, S.L., 2024. Multiple View Geometry Transformers for 3D Human Pose Estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 708–717. Available at: `https://github.com/XunshanMan/MVGFormer` (Accessed: 5 April 2025).

[2] Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F. and He, Z., 2021. Deep 3D Human Pose Estimation: A Review. *Computer Vision and Image Understanding*, 210, 103225. Available at: `https://doi.org/10.1016/j.cviu.2021.103225` (Accessed: 5 April 2025).

[3] Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J. and Yuan, J., 2019. Hand Pose Estimation: A Survey. *Computer Vision and Image Understanding*, 190, 102626. Available at: `https://doi.org/10.1016/j.cviu.2019.102626` (Accessed: 5 April 2025).

[4] Garcia-Hernando, G., Yuan, S., Baek, S. and Kim, T.K., 2019. A Survey on 3D Hand Pose Estimation: Cameras, Methods, and Datasets. *Computer Vision and Image Understanding*, 192, 102897. Available at: `https://doi.org/10.1016/j.cviu.2019.102897` (Accessed: 5 April 2025).

[5] Chen, X., Zhan, H., Zhou, Y. and Wang, R., 2022. Efficient Annotation and Learning for 3D Hand Pose Estimation: A Survey. *Journal of Visual Communication and Image Representation*, 85, 103572. Available at: `https://doi.org/10.1016/j.jvcir.2022.103572` (Accessed: 5 April 2025).

[6] Palmero, C., Escalera, S., Subramanian, R., Bulling, A., Baró, X. and Pantic, M., 2021. The UDIVA Dataset: Multimodal Interactions in Dyadic Settings. Available at: `https://chalearnlap.cvc.uab.cat/dataset/41/description/` (Accessed: 5 April 2025).

[7] Palmero, C., Baró, X., Subramanian, R., Bulling, A., Pantic, M. and Escalera, S., 2022. Understanding Social Behavior in Dyadic and Small Group Interactions. In: *Proceedings of the Chalearn Looking at People Workshop – DYAD Track*, PMLR 173:4–52. Available at: `https://chalearnlap.cvc.uab.cat/dataset/41/description/` (Accessed: 5 April 2025).

[8] Boukhayma, A., Bem, R.d. and Torr, P.H., 2019. 3D Hand Shape and Pose from Images in the Wild. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10843–10852.

[9] Yang, J., Li, J., Li, G., Wu, H.Y., Shen, Z. and Fan, Z., 2024. MLPHand: Real Time Multi-View 3D Hand Reconstruction via MLP Modeling. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Avail-

able at: https://github.com/jackyyang9/MLPHand (Accessed: 6 April 2025).

[10] Remelli, E., Han, S., Honari, S., Fua, P. and Wang, R., 2020. Lightweight Multi-View 3D Pose Estimation through Camera-Disentangled Representation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6040–6049.

[11] Palmero, C., Barquero, G., Jacques Junior, J.C.S., Clapés, A., Núñez, J., Curto, D., Smeureanu, S., Selva, J., Zhang, Z., Saeteros, D., Gallardo-Pujol, D., Guilera, G., Leiva, D., Han, F., Feng, X., He, J., Tu, W.-W., Moeslund, T.B., Guyon, I. and Escalera, S., 2022. ChaLearn LAP Challenges on Self-Reported Personality Recognition and Non-Verbal Behavior Forecasting During Social Dyadic Interactions: Dataset, Design, and Results. *Proceedings of Machine Learning Research*, 173:4–52. Available at: https://chalearnlap.cvc.uab.cat/dataset/41/description/ (Accessed: 6 April 2025).

[12] Ali, M.A., Robertini, N. and Stricker, D., 2025. HandMvNet: Real-Time 3D Hand Pose Estimation Using Multi-View Cross-Attention Fusion. In: *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) – Volume 2: VISAPP*, pp. 555–562. Available at: https://doi.org/10.5220/0013107300003912 (Accessed: 6 April 2025).

[13] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. and Grundmann, M., 2020. MediaPipe Hands: On-device Real-time Hand Tracking. Google Research. Available at: https://mediapipe.dev (Accessed: 6 April 2025).

[14] Hartley, R. and Zisserman, A., 2004. *Multiple View Geometry in Computer Vision*. 2nd edn. Cambridge University Press.

[15] Triggs, B., McLauchlan, P.F., Hartley, R.I. and Fitzgibbon, A.W., 2000. Bundle Adjustment – A Modern Synthesis. In: Vision Algorithms: Theory and Practice, pp. 298–372.

[16] Stachniss, C., 2020. Bundle Adjustment – Part I: Lecture Notes, Photogrammetry and Robotics Lab, University of Bonn. Available at: https://www.ipb.uni-bonn.de/lectures/ba-ws20/ (Accessed: 6 April 2025).

[17] Liao, Z., Zhu, J., Wang, C., Hu, H. and Waslander, S.L., 2024. Multiple View Geometry Transformers for 3D Human Pose Estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 708–717. Available at: https://github.com/XunshanMan/MVGFormer (Accessed: 6 April 2025).

Appendix

Example 2D extraction

Listing 1: Hand and Pose Keypoint Extraction Script

```
import cv2
import mediapipe as mp
import pandas as pd
import numpy as np
import os


v = {
    "c1": video paths,
    "c2": video paths"
}
o = f"{b}/error"
os.makedirs(o, exist_ok=True)

mp_pose = mp.solutions.pose
mp_hands = mp.solutions.hands
p = mp_pose.Pose(False, 0.5, 0.5)
h = mp_hands.Hands(False, 4, 0.5)

def extract(vp, label):
    cap = cv2.VideoCapture(vp)
    idx = 0
    out_data = []

    while cap.isOpened() and idx < 1000:
        ret, frame = cap.read()
        if not ret:
            break
        h_, w_ = frame.shape[:2]
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        r = p.process(rgb)
        s = []
        if r.pose_landmarks:
            lms = r.pose_landmarks.landmark
            try:
                l = np.array([lms[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x *
                              lms[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y *
                r = np.array([lms[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x *
                              lms[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y *
                m = (l + r) / 2
```

```
                s.append(m)
            except:
                pass

        hands_result = h.process(rgb)
        if hands_result.multi_hand_landmarks and s:
            for i, (hl, hd) in enumerate(zip(
                hands_result.multi_hand_landmarks,
                hands_result.multi_handedness)):

                hlab = hd.classification[0].label
                conf = hd.classification[0].score
                wrist = hl.landmark[0]
                wrist_xy = np.array([wrist.x * w_, wrist.y * h_])
                dists = [np.linalg.norm(wrist_xy - sh) for sh in s]
                pid = np.argmin(dists)
                mid = f"{idx}_{i}"

                for j, lm in enumerate(hl.landmark):
                    out_data.append({
                        "frame_index": idx,
                        "person_id": pid,
                        "hand_type": hlab,
                        "landmark_index": j,
                        "x": int(lm.x * w_),
                        "y": int(lm.y * h_),
                        "confidence": conf,
                        "matched_id": mid,
                        "camera": label
                    })

        idx += 1
    cap.release()
    return pd.DataFrame(out_data)

df1 = extract(v["c1"], "cam1")
df2 = extract(v["c2"], "cam2")
df1.to_csv(getting csv with the path)
df2.to_csv(getting cvs with the path)
```

tringulation example code

Listing 2: Bundle-adjusted triangulation of hand landmarks

```
import pandas as pd
import numpy as np
import cv2
```

```
import os
from scipy.optimize import least_squares

a = "-input-path"
b = "-input-path"
o = "output-path"

c = np.array([[1717.75, 0, 622.75], [0, 1537.85, 394.49], [0, 0, 1]])
d = np.array([-0.3789, 0.2004, 0, 0, 0])
c2 = np.array([[1699.26, 0, 653.08], [0, 1521.33, 388.14], [0, 0, 1]])
d2 = np.array([-0.3815, 0.2191, 0, 0, 0])
r = np.array([[0.06783436, 0.16412933, -0.98410369],
              [-0.1399054, 0.97819436, 0.15350007],
              [0.98783854, 0.12726884, 0.08931777]])
t = np.array([[1335.7205], [-177.9507], [1135.2378]])

p1 = c @ np.hstack((np.eye(3), np.zeros((3, 1))))
p2 = c2 @ np.hstack((r, t))

def undistort(x, k, d):
    return cv2.undistortPoints(np.expand_dims(x, axis=1), k, d, None, k).squeeze

def tri(x1, x2):
    x1, x2 = x1.T, x2.T
    p = cv2.triangulatePoints(p1, p2, x1, x2)
    return (p[:3] / p[3]).T

def err(x, a1, a2, c, d, c2, d2, r, t):
    x = x.reshape(-1, 3)
    p1_, _ = cv2.projectPoints(x, np.zeros(3), np.zeros(3), c, d)
    p2_, _ = cv2.projectPoints(x, cv2.Rodrigues(r)[0], t, c2, d2)
    return np.concatenate([(a1 - p1_.squeeze()).ravel(), (a2 - p2_.squeeze()).ra

df1 = pd.read_csv(a)
df2 = pd.read_csv(b)
thresh = 0.6

g1 = df1.groupby("frame_index")
g2 = df2.groupby("frame_index")

res = []
for f in sorted(set(g1.groups) & set(g2.groups)):
    d1 = g1.get_group(f)
    d2 = g2.get_group(f)

    ids = set(d1["matched_id"].unique()) & set(d2["matched_id"].unique())
```

```
    for m in ids:
        h1 = d1[(d1["matched_id"] == m) & (d1["confidence"] >= thresh)]
        h2 = d2[(d2["matched_id"] == m) & (d2["confidence"] >= thresh)]

        if len(h1) != 21 or len(h2) != 21:
            continue

        a1 = h1[["x", "y"]].values.astype(np.float32)
        a2 = h2[["x", "y"]].values.astype(np.float32)

        a1u = undistort(a1, c, d)
        a2u = undistort(a2, c2, d2)

        x0 = tri(a1u, a2u)
        opt = least_squares(err, x0.ravel(), method='lm', args=(a1, a2, c, d, c2
        x3d = opt.x.reshape(-1, 3)

        for i, pt in enumerate(x3d):
            res.append({
                "frame_index": f,
                "matched_id": m,
                "landmark_index": i,
                "X": pt[0],
                "Y": pt[1],
                "Z": pt[2]
            })

os.makedirs(os.path.dirname(o), exist_ok=True)
pd.DataFrame(res).to_csv(o, index=False)
```

example for mediahands holistic visualisation

Listing 3: Holistic keypoint visualisation with MediaPipe

```
import cv2
import pandas as pd
import numpy as np
import os

csv_path = "csvpath-ie-3d-extracted-one"
v_path = "videopath"
out_path = "output-"

df = pd.read_csv(csv_path)

cap = cv2.VideoCapture(v_path)
fps = cap.get(cv2.CAP_PROP_FPS)
```

```
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

out = cv2.VideoWriter(out_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))

pose_colour = (0, 255, 255)
left_colour = (0, 255, 0)
right_colour = (0, 0, 255)

pose_pts = {'LEFT_SHOULDER', 'RIGHT_SHOULDER', 'NOSE'}

i = 0
while i < 1000:
    ret, f = cap.read()
    if not ret:
        break

    d = df[df['frame_index'] == i]

    for _, row in d.iterrows():
        x, y = int(row['x']), int(row['y'])
        t = row['landmark_type']

        if t == 'pose' and row['name'] in pose_pts:
            cv2.circle(f, (x, y), 6, pose_colour, -1)
            cv2.putText(f, row['name'], (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.
        elif t == 'hand':
            c = left_colour if row['hand_type'] == 'Left' else right_colour
            cv2.circle(f, (x, y), 4, c, -1)

    cv2.putText(f, f"Frame {i}", (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 25
    out.write(f)
    i += 1

cap.release()
out.release()
```

Listing 4: Holistic Keypoint Visualisation Script

```
import cv2
import pandas as pd
import numpy as np
import os

csv_path = "csvpath ie 3d extracted one"
v_path = "videopath"
out_path = "output "
```

```
df = pd.read_csv(my input path for the extracted values)

cap = cv2.VideoCapture(video_path)
fps = cap.get(cv2.CAP_PROP_FPS)
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_

POSE_COLOUR = (0, 255, 255)
HAND_LEFT_COLOUR = (0, 255, 0)
HAND_RIGHT_COLOUR = (0, 0, 255)

pose_indices = {'LEFT_SHOULDER', 'RIGHT_SHOULDER', 'NOSE'}

frame_idx = 0
while frame_idx < 1000:
    ret, frame = cap.read()
    if not ret:
        break

    df_frame = df[df['frame_index'] == frame_idx]

    for _, row in df_frame.iterrows():
        x, y = int(row['x']), int(row['y'])
        l_type = row['landmark_type']

        if l_type == 'pose' and row['name'] in pose_indices:
            cv2.circle(frame, (x, y), 6, POSE_COLOUR, -1)
            cv2.putText(frame, row['name'], (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX
        elif l_type == 'hand':
            colour = HAND_LEFT_COLOUR if row['hand_type'] == 'Left' else HAND_RI
            cv2.circle(frame, (x, y), 4, colour, -1)

    cv2.putText(frame, f"Frame {frame_idx}", (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
    out.write(frame)
    frame_idx += 1

cap.release()
out.release()
```