RGB-D CAMERAS & BODY TRACKING

Virtual Reality Course — Updated 2025/2026

From depth sensing to pose estimation, skeletons, and avatar animation

eleonora.chitti@unimi.it

Laboratorio Realtà Virtuale 2025/2026

IMPORT CHARACTERS AND ANIMATIONS IN UNITY

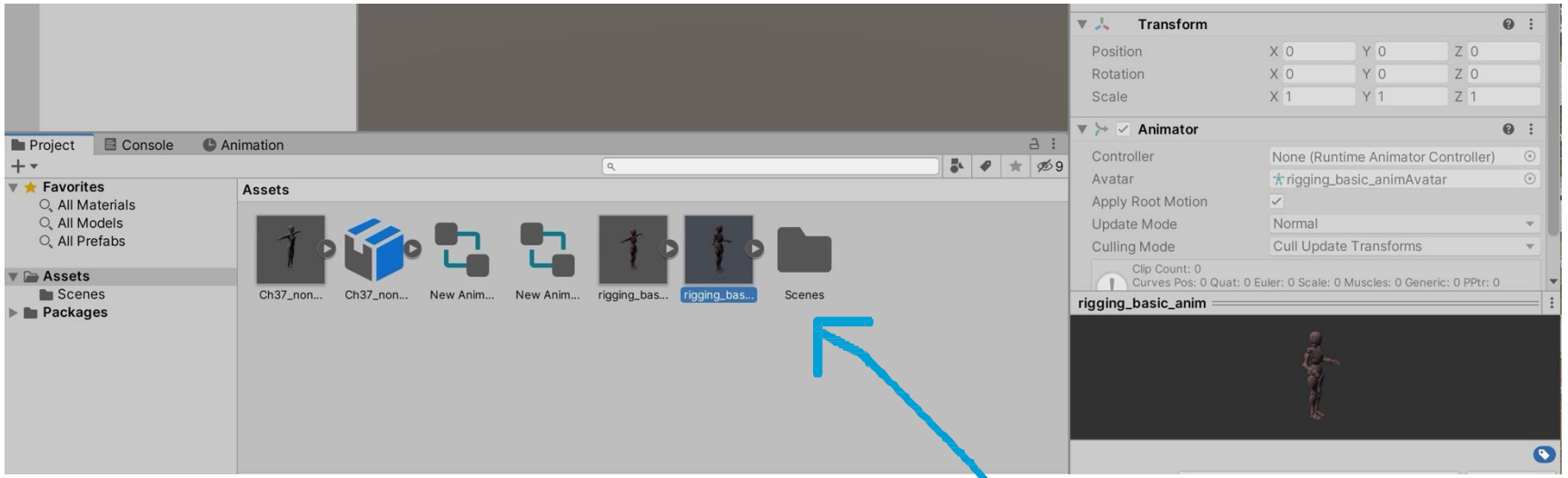
UNITY ANIMATION AND RIGGING

- A rig consists in a series of joints mimicking human bones; the character's mesh is connected to the bones (with the weight paint) and moves together with them.

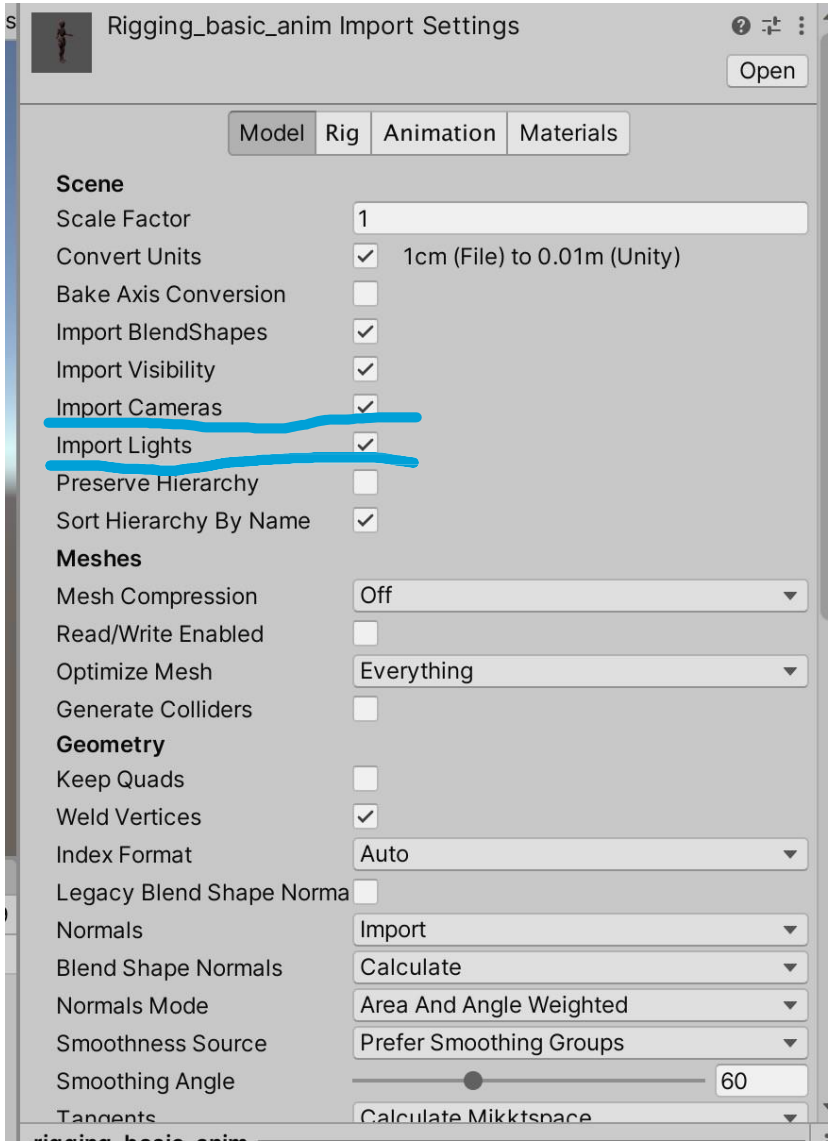
- In Unity the rigged and skin properties need to be setup properly when importing an FBX cheater rigged with a tool as Blender or Maya.

CHARACTER'S SETUP

1. Import the FBX by dragging it into the ProjectWindow of the Unity Project
2. Select it
3. In the Inspector panel, click on the Model tab and uncheck the Import camera and Import light (we didn't exported the camera and light in our FBX) **– optional , you can skip this step–**
4. **If you have performed the step 3** then click on "Apply" button



OPTIONAL (STEPS 3,4)



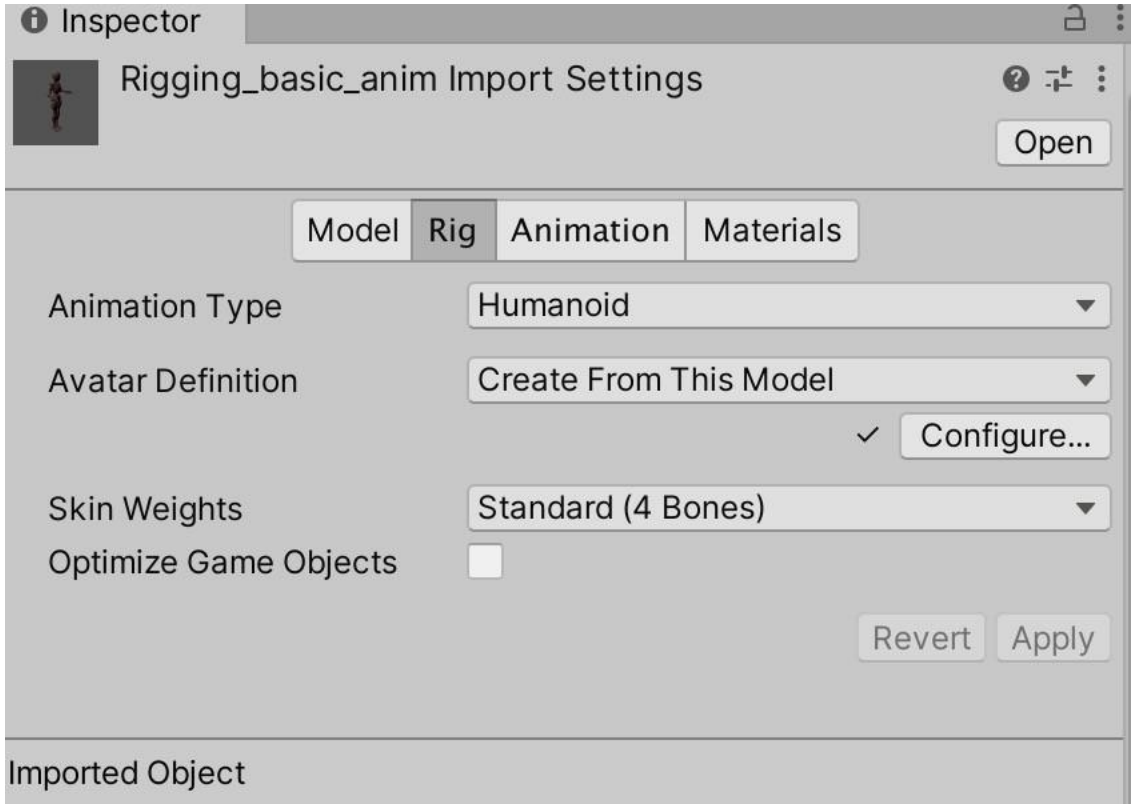
3

Scroll down to find theApply button

4

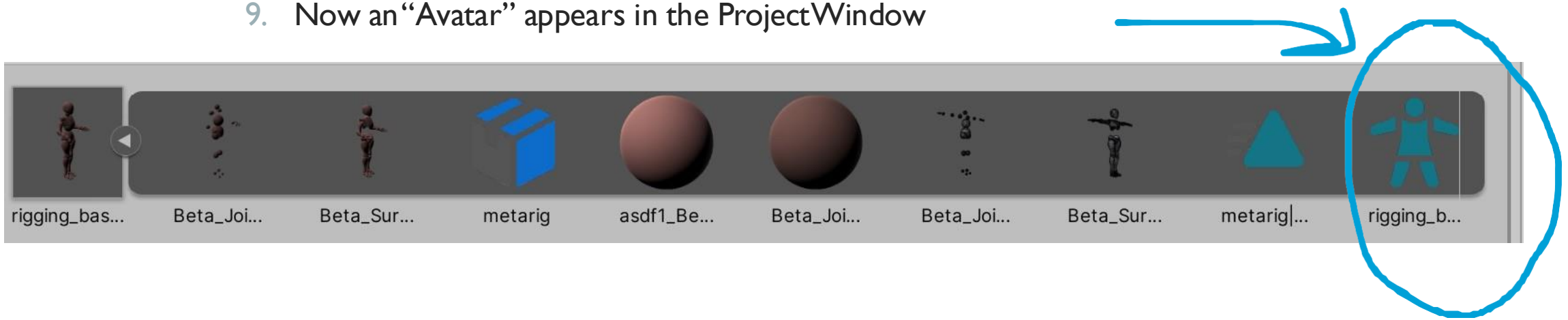
CHARACTER'S SETUP

5. Now click on Rig
6. Change the Animation Type from Legacy to Humanoid
7. Change Avatar Definition to Create from this model
8. Apply



CHARACTER'S SETUP

9. Now an "Avatar" appears in the ProjectWindow

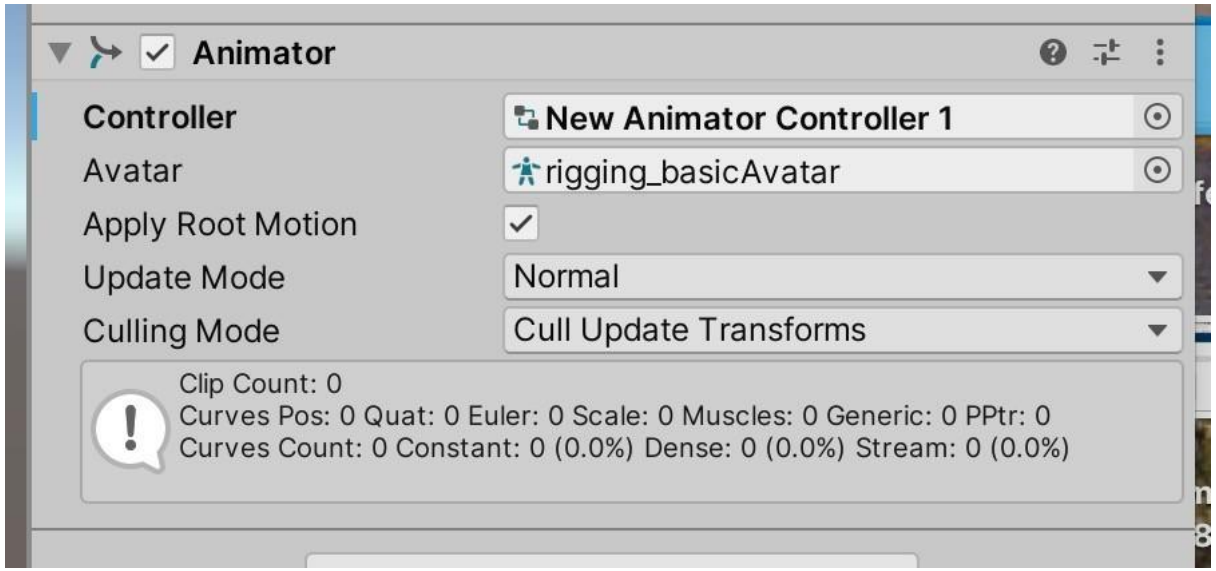


SETUP ANIMATION

- Import the FBX by dragging it into the ProjectWindow of the Unity Project
- Select this fbx in the Project window
- Repeat all the steps from 2 to 9 (slides 20,21 and 22)

ADD ANIMATOR

- In the ProjectWindow right click to add a newAnimator Controller
- Double click on the icon to open it
- Drag the FBX with the animation inside,it will be automatically bind as a state in theAnimator Finite State Machine
- Add the animator to the avatar Game Object



PLAY IN THE EDITOR

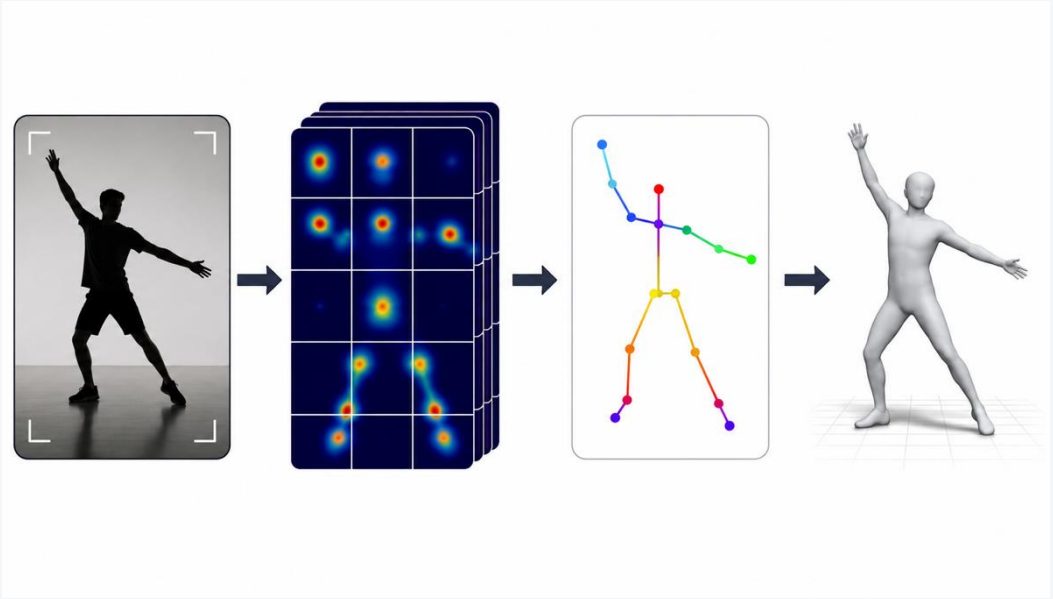
- Now you can play and check the results!

PART 1 — RGB-D CAMERAS

Depth sensing as a source of 3D information for interaction and tracking

LEARNING GOALS

- Understand the difference between RGB, depth, RGB-D and LiDAR sensing.
- Explain body tracking as a sequence of detection, pose estimation and temporal tracking.
- Compare 3D tracking from depth sensors with 2D pose estimation from standard cameras.
- Relate skeleton landmarks to Unity humanoid rigs and avatar animation.
- Identify common failure cases and practical constraints in VR/AR applications.



Pose estimation: image → heatmaps → keypoints → avatar

RGB-D CAMERA

AI-generated illustration of RGB, depth and point cloud streams

RGB stream

A conventional color image: width × height × 3 color channels.

Depth stream

Each pixel stores an estimate of distance from the camera, usually in millimeters or meters.

RGB-D stream

Color and depth are spatially aligned, enabling 3D reconstruction, segmentation and body tracking.

Core idea: depth gives the missing Z dimension.

HOW DEPTH IS MEASURED

Structured light

Project a known IR pattern; infer depth from deformation.

Time of Flight

Measure the time/phase of reflected light.

Active stereo

Two IR cameras compare left/right images with IR texture.

LiDAR

Laser pulses estimate distance and build dense spatial maps.

All methods estimate distance indirectly. Accuracy depends on sensor design, reflectivity, lighting, distance, field of view, calibration and motion.

Depth is not magic: it is a measurement with noise and missing values

DEPTH DATA QUALITY

Lighting

Strong sunlight can saturate IR sensors.

Distance

Too close or too far reduces precision.

Materials

Black, shiny, glass or transparent surfaces can fail.

Shape

Thin or complex objects are hard to reconstruct.

Occlusion

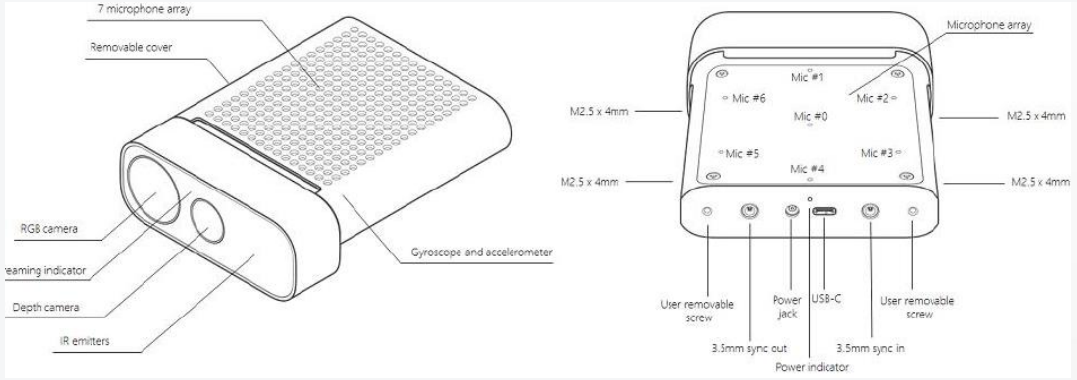
Hidden body parts cannot be directly observed.

Calibration

RGB/depth misalignment affects skeleton fitting.

In body tracking, noisy depth causes jitter, incorrect joints and unstable avatar motion.

DEPTH CAMERA DEVICES



Azure Kinect DK hardware diagram

Kinect family

Accessible RGB-D tracking ecosystem; Kinect v1/v2 discontinued; Azure Kinect DK moved to partner ecosystem.

RealSense D400

Stereo depth cameras used in robotics, drones, prototyping and spatial perception.

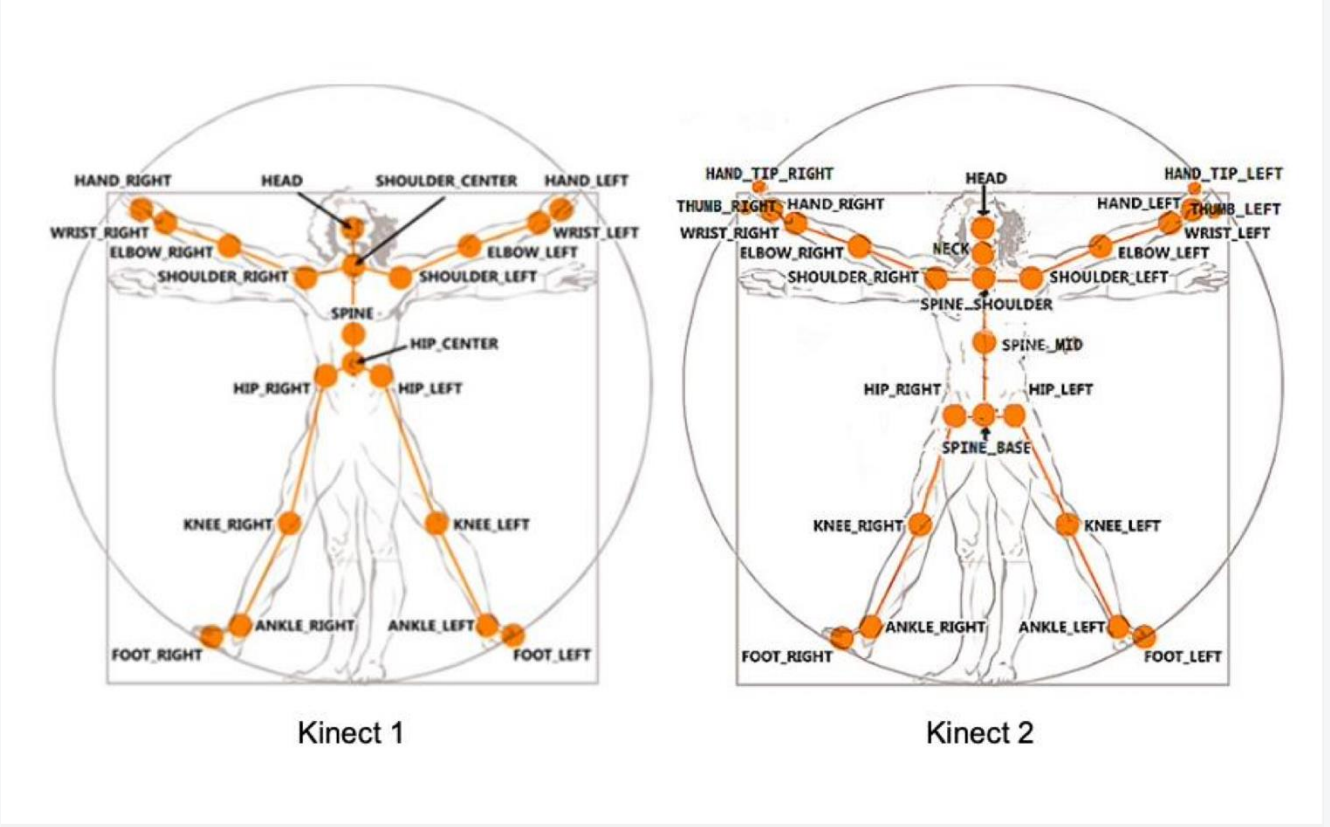
Mobile depth

LiDAR and TrueDepth enable AR, face tracking, scene reconstruction and occlusion.

For body tracking, device choice affects range, latency, OS support, SDK availability and integration workflow.

KINECT: WHY IT MATTERED

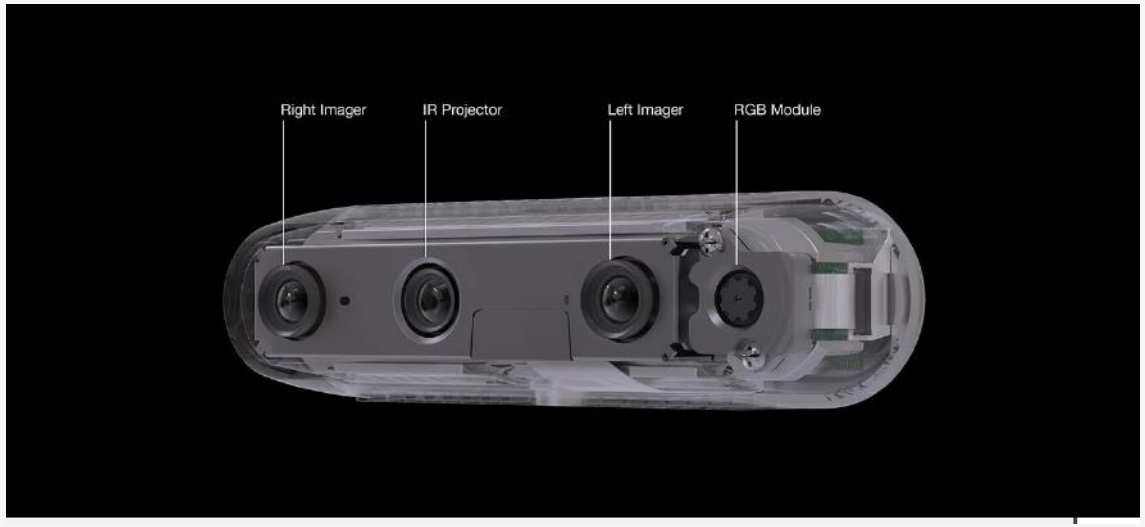
- Kinect made markerless full-body tracking widely accessible outside specialized labs.
- It combined RGB, depth, infrared and SDK-level skeleton estimation.
- Kinect v1/v2 became common in exergames, rehabilitation, HCI and VR prototyping.
- Azure Kinect improved the sensor stack, but long-term hardware support shifted away from Microsoft devices.



Kinect 1 vs Kinect 2 skeleton models

Historical relevance: it shaped today's expectations for natural interaction

REALSENSE D435



RealSense D435 internal layout from previous course slides

Stereo depth

Depth is generated by comparing left and right infrared images.

Compact form factor

Small and lightweight, useful for robots, mobile devices and interactive installations.

SDK ecosystem

C++, Python, C#, wrappers and integrations for Unity/Unreal-oriented workflows.

Important: RealSense provides depth. Body tracking still requires a separate algorithm or middleware.

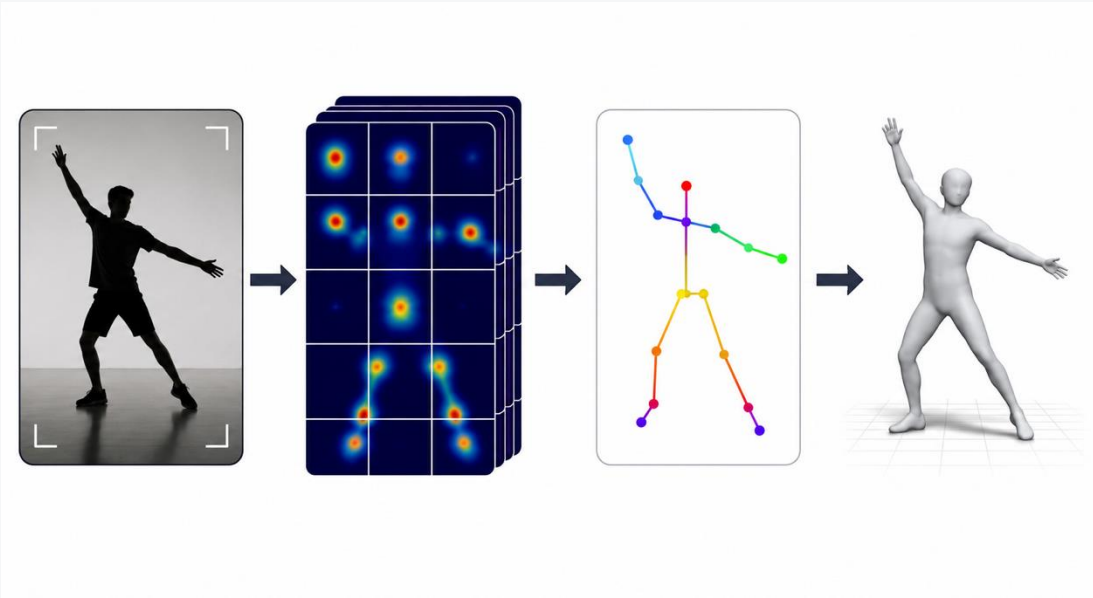
PART 2 — BODY TRACKING

From sensor streams to skeletons, joints and motion over time

WHAT IS BODY TRACKING?

Body tracking estimates the position and movement of a person's body over time.

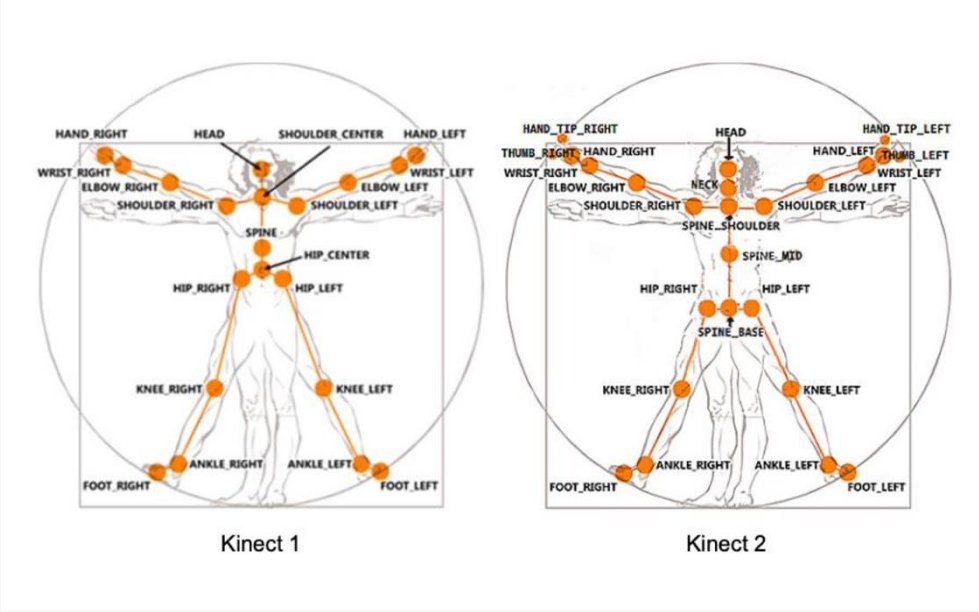
- The body is represented by a skeleton model.
- The skeleton is a graph of keypoints connected by bones.
- Each keypoint usually has position, confidence and sometimes orientation.
- Tracking adds temporal consistency: the same body and joints across frames.



Pose estimation pipeline

Image/video → body model → tracked motion

SKELETONS, JOINTS AND BONES



Skeleton landmark examples

Joint / landmark

A semantic body point such as shoulder, elbow, hip or knee.

Bone

A connection between two joints, used to create a kinematic chain.

Pose

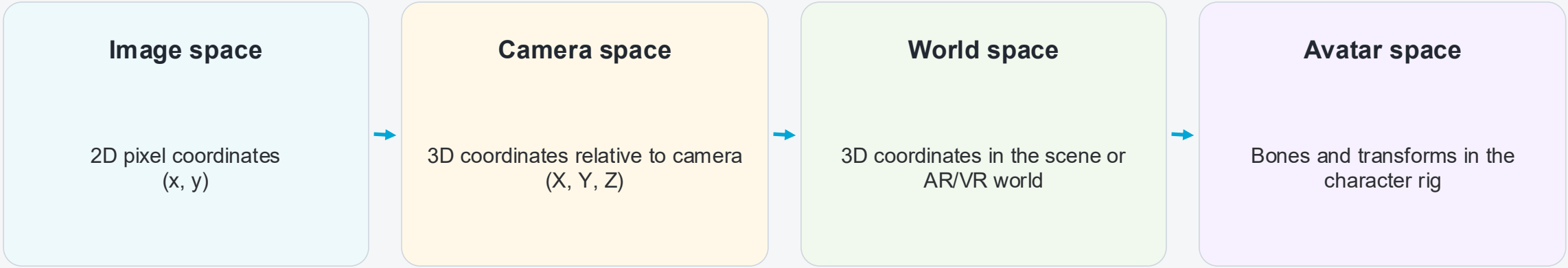
The set of joint positions at a single frame or instant.

Motion

A temporal sequence of poses: pose(t), pose(t+1), ...

Different systems use different skeleton definitions: Kinect v1/v2, OpenPose, ML Kit, MediaPipe, Nuitrack and ARKit do not expose the same joints.

COORDINATE SPACES



Body tracking often requires coordinate conversion, scaling, smoothing and retargeting.

- 2D methods output pixel landmarks and confidence scores.
- 3D methods output joints in metric or pseudo-metric space.
- Unity transforms require a consistent origin, orientation and scale.

GENERIC BODY TRACKING PIPELINE

Input

RGB, depth, IR or video frames

Person detection

Find body region or crop

Pose estimation

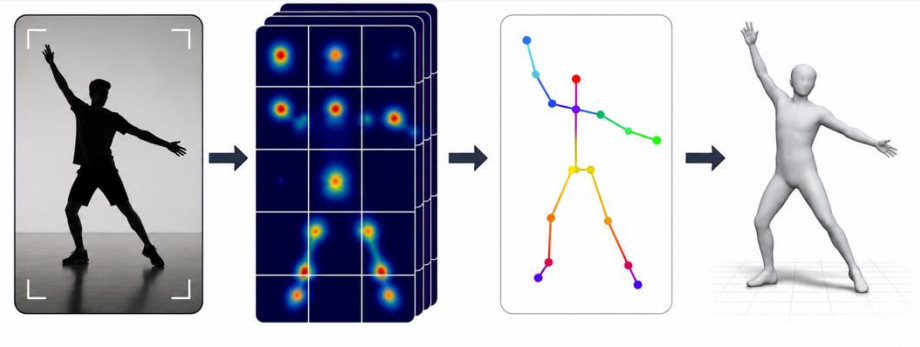
Predict landmarks and confidence

Tracking

Stabilize identity and motion over time

Application

Avatar, gesture, analytics, interaction



- Real systems add confidence thresholds, outlier rejection and temporal smoothing.
- For avatar animation, raw joint positions are converted into bone rotations.
- Latency matters: delay breaks presence in immersive applications.

3D BODY TRACKING FROM DEPTH

Depth provides geometric cues

- Depth cameras produce a dense map of distances.
- The person can be segmented from the background using geometry.
- A skeleton model is fitted to the observed 3D body surface or depth silhouette.
- The output can be metric 3D joint positions relative to the camera.

Strength

True spatial information

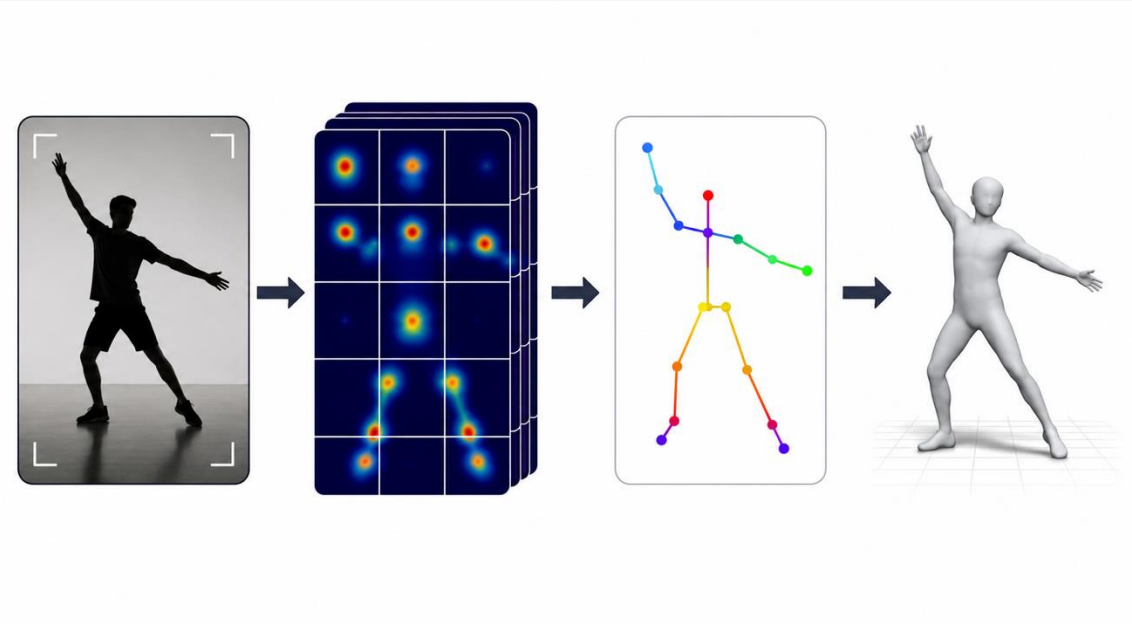
Weakness

Sensor-specific constraints

Use cases

VR, rehab, robotics, installations

2D POSE ESTIMATION FROM RGB



A learned model predicts keypoints from pixels

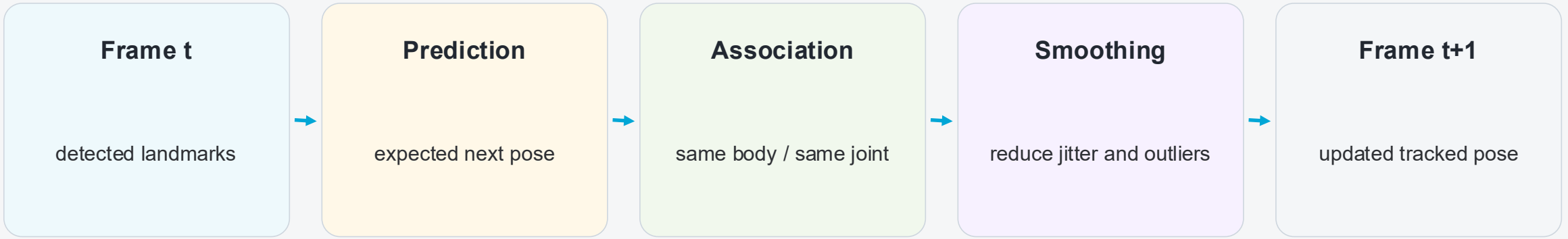
- Uses a standard camera or video stream: no depth sensor required.
- Deep learning predicts semantic landmarks: nose, shoulders, elbows, wrists, hips, knees, ankles, etc.
- Some systems estimate a relative Z value or lift 2D joints into 3D using learned priors.
- It is flexible and scalable, but 3D ambiguity remains a core problem.

Key limitation: many 3D poses can produce similar 2D projections.

2D VS 3D TRACKING

Aspect	2D pose estimation	3D body tracking
Input	RGB image/video	Depth/RGB-D, multi-view, LiDAR or 3D model
Output	2D keypoints + confidence	3D joints in camera/world space
Hardware	Any camera, phone or webcam	Depth sensor, AR device or more complex setup
Strength	Cheap, accessible, easy to deploy	Metric geometry, spatial interaction
Weakness	Depth ambiguity and occlusion	Hardware limits, sensor noise, support issues

TRACKING OVER TIME



Jitter

Small frame-to-frame fluctuations that make avatars vibrate.

Identity switches

In multi-person scenes, the tracker may swap users.

Latency

Smoothing improves stability but may increase delay.

Good body tracking is both spatially accurate and temporally stable.

PART 3 — METHODS AND TOOLS

A concise overview of common SDKs and body tracking ecosystems

OPENPOSE

Main idea

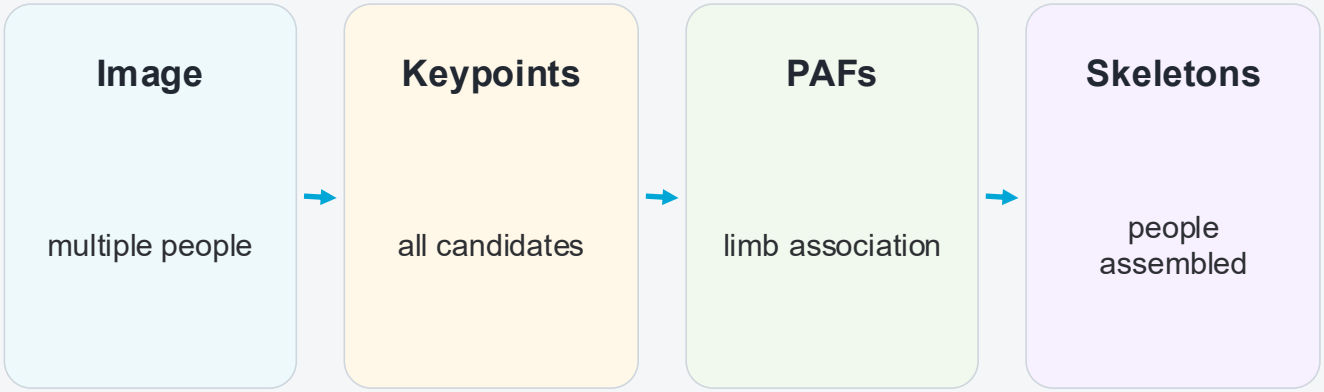
Bottom-up multi-person pose estimation: detect all keypoints, then assemble people.

Key contribution

Part Affinity Fields model the relationship between body parts.

Output

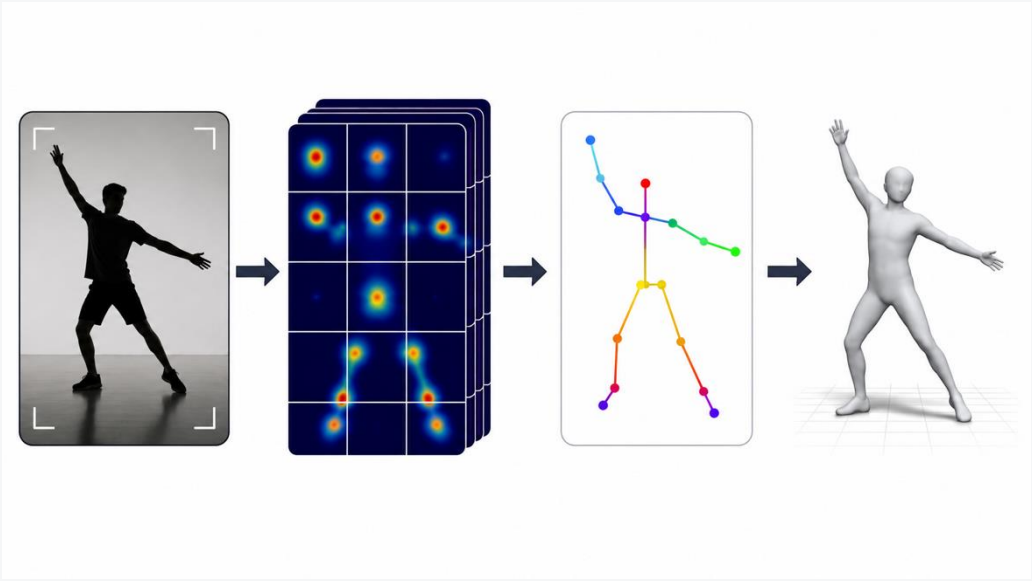
Body, foot, hand and face keypoints; commonly cited as 135 keypoints.



OpenPose remains a reference point for understanding multi-person 2D pose estimation.

GOOGLE ML KIT AND MEDIAPIPE

- ML Kit Pose Detection targets real-time app development from continuous video or static images.
- MediaPipe Pose / Pose Landmarker provides body landmarks for image and video tasks.
- MediaPipe is designed for on-device, lightweight, cross-platform inference.
- Typical output includes 33 body landmarks, with image coordinates and 3D/world coordinates depending on the API.

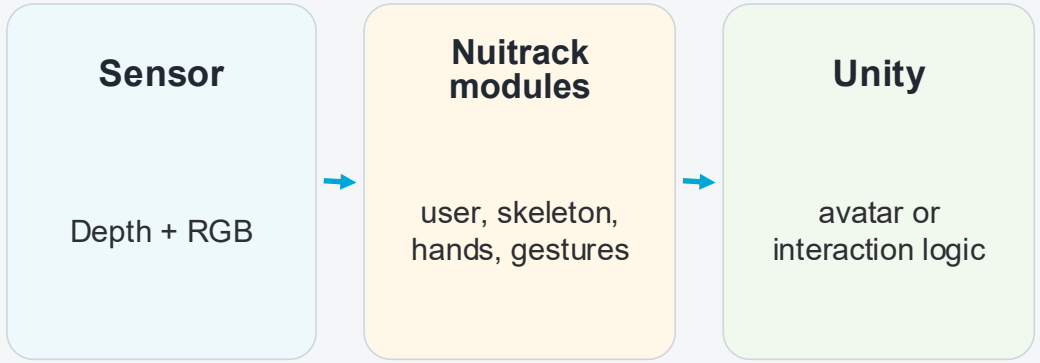


Modern learned pose pipeline

MediaPipe will be used in the upcoming practical exercise

NUITRACK

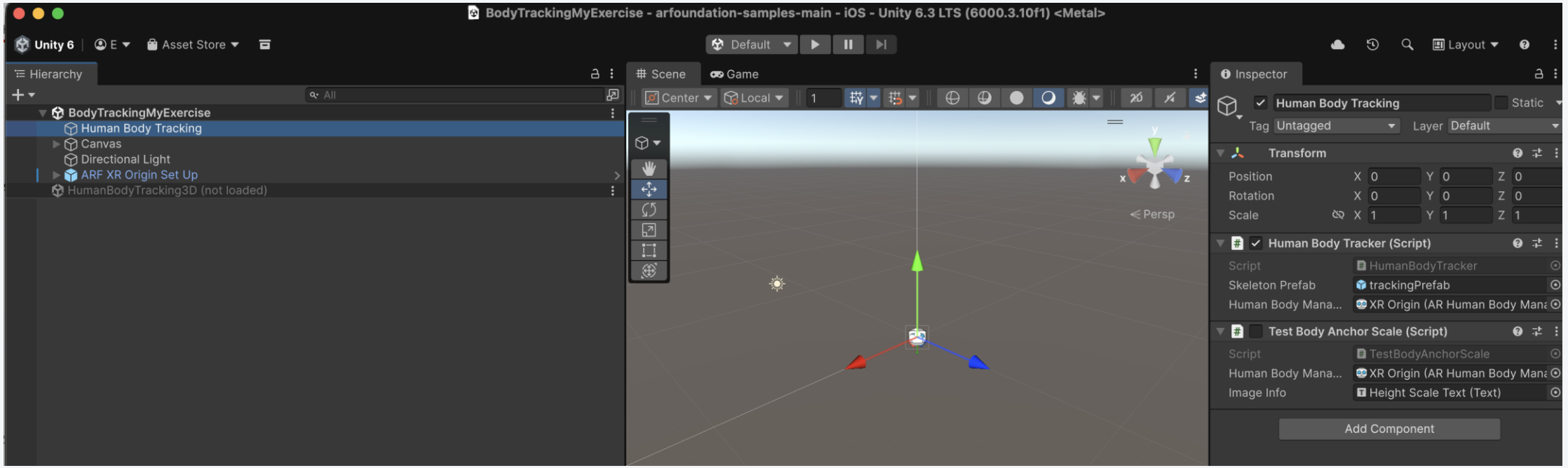
- 3D body / skeletal tracking middleware for depth sensors and RGB-D devices.
- Targets gesture recognition and skeleton tracking for Windows, Linux and Android workflows.
- Useful when the sensor provides depth but not a built-in body tracking SDK.
- Unity integration is available through its SDK and examples.



Practical note

Check device compatibility, license requirements, OS support and Unity version before planning the lab.

ARKIT (iOS) 3D BODY TRACKING



Unity AR Foundation body tracking setup

- ARKit supports 3D body motion capture on compatible iOS/iPadOS devices.
- The skeleton can drive a virtual character in an AR scene.
- Unity AR Foundation exposes body tracking through AR Human Body Manager.
- Device and OS requirements are strict; body tracking is ARKit-only in AR Foundation samples.

Mobile AR: body tracking + world tracking + camera passthrough

TRACKING ECOSYSTEM AT A GLANCE

System	Input	Typical role
OpenPose	RGB	Research/reference 2D multi-person pose
ML Kit	RGB	Mobile app pose detection
MediaPipe	RGB	Real-time lightweight pose landmarker
Nuitrack	RGB-D / depth	Middleware for skeletal tracking
ARFoundation	(ARKit (iOS camera + sensors) for 3D ARCore/Arkit for 2D body	AR body motion capture
RealSense	Depth/RGB-D	Sensing hardware; tracking via algorithms

PART 4 — FROM SKELETON TO AVATAR

How tracking data becomes motion inside a VR/AR application

RETARGETING: SKELETON TO AVATAR

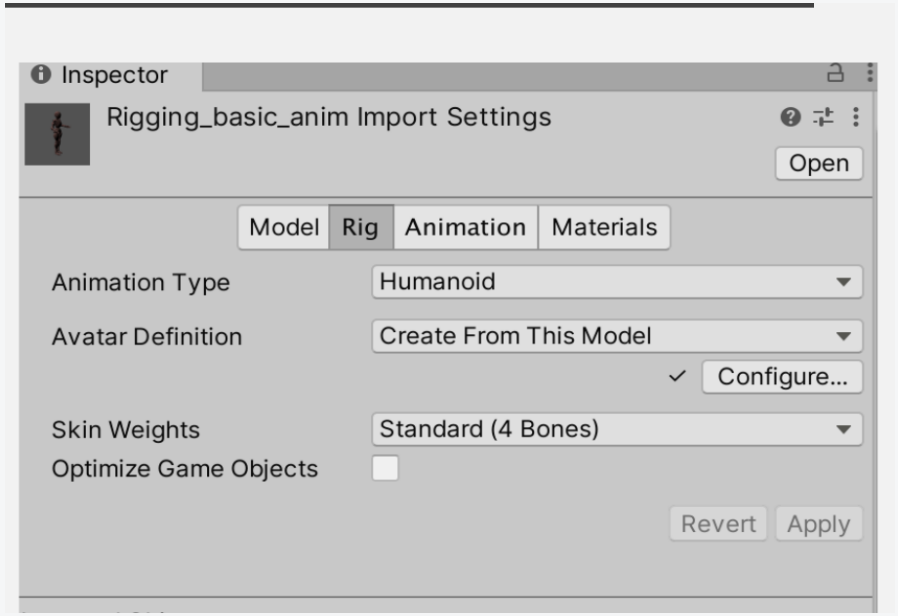
Motion capture data drives a virtual avatar

- Tracking skeletons and character rigs rarely have the same bone hierarchy.
- Retargeting maps source joints to target bones.
- A humanoid avatar usually needs bone rotations rather than raw joint positions.
- Calibration is needed for scale, T-pose/A-pose alignment and coordinate axes.

Goal: make the avatar move naturally while respecting the rig constraints.

UNITY HUMANOID RIG RECAP

- Import a rigged FBX character into the Unity project.
- Set Animation Type to Humanoid when the model is human-like.
- Create or validate the Avatar definition.
- Check material and texture import settings if the model appears gray.
- Tracking scripts will later update transforms or feed animation rigging constraints.



Humanoid import settings from previous course slides

AVATAR ANIMATION WORKFLOW

Calibration pose

A known pose aligns the user and avatar skeletons.

Bone lengths

Source and target bodies often have different proportions.

IK constraints

Inverse kinematics can keep feet, hands and head stable.

In VR, avatar motion is judged perceptually: stable and plausible often matters more than numerically perfect.

APPLICATIONS IN VR/AR AND BEYOND

Immersive avatars

Full-body presence, social VR and embodiment.

Exergames

Exercise, scoring, feedback and safety.

Rehabilitation

Range of motion, therapy compliance, remote monitoring.

Robotics

Human following, gesture commands, safety zones.

Sports analysis

Pose metrics, technique assessment, coaching.

Film & animation

Low-cost motion capture and previsualization.

COMMON CHALLENGES

Real-time tracking in a room-scale setup

- **Occlusion**
- **Ambiguity (is a body shape?)**
- **Domain shift**
- **Performance**
- **Calibration**

Failure handling is part of the design: confidence thresholds, fallbacks and clear user feedback.

HOW DO WE EVALUATE TRACKING?

Accuracy
How close are predicted joints to ground truth?

Robustness
Does tracking survive occlusion and difficult poses?

Latency
How long between movement and visible response?

Stability
Is the skeleton smooth or jittery?

Usability
Can users set it up reliably in the lab or at home?

Privacy
Can data stay on-device and avoid storing raw video?

For teaching labs, reliability and repeatability are just as important as state-of-the-art accuracy.

ETHICS, PRIVACY AND SAFETY

- Body tracking data can reveal health, disability, identity and behavioral patterns.
- Prefer on-device processing when possible, especially for prototypes.
- Avoid storing raw video unless it is necessary, consented and protected.
- Explain limitations clearly: tracking can be biased by camera view, body shape, clothing and mobility differences.
- In VR/AR, inaccurate tracking can cause discomfort, and unsafe movements.

Design principle

Track only what the application truly needs.

Course principle

Use transparent, reproducible pipelines.

User principle

Give feedback when tracking confidence is low.

UPCOMING LABORATORY: MEDIAPIPE POSE

Placeholder for the practical exercise

To be added later

Setup instructions, code and dataset/video examples.

Possible goals

Landmark visualization, angle computation, posture classification.

Optional challenge

Retarget detected motion to a Unity avatar.

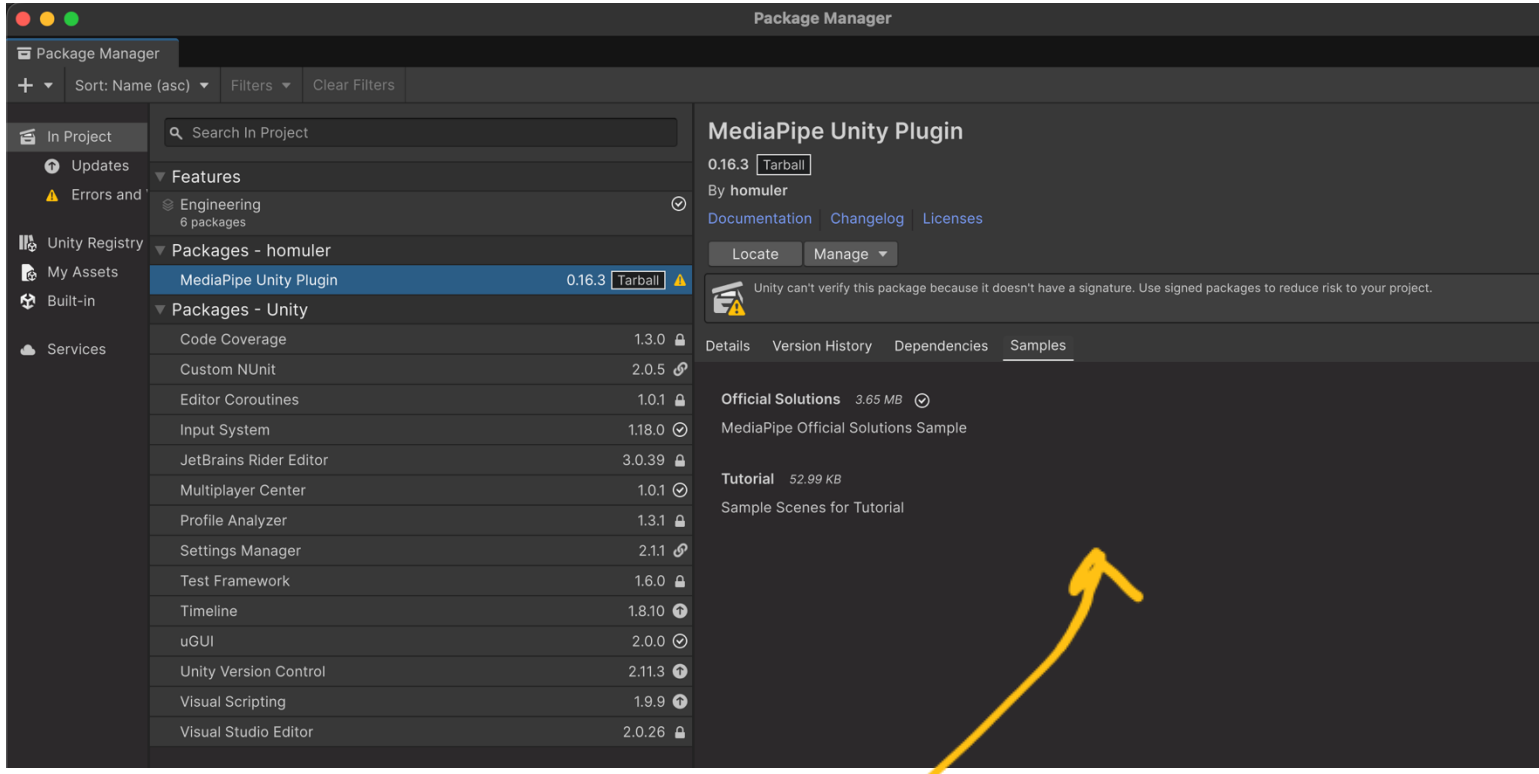
Donwload MadiaPipe from here: https://github.com/homuler/MediaPipeUnityPlugin

UPCOMING LABORATORY: MEDIAPIPE POSE

- Import the com.github.homuler.mediapipe-*.tgz tarball package from: https://github.com/homuler/MediaPipeUnityPlugin/releases

- Install the tgz file in Unity from the Package Manager following the instructions here: https://docs.unity3d.com/Manual/upm-ui-tarball.html

- Once installed, in the Package Manager go to MediaPipe > Samples and download the Official Solutions



Donwload MadiaPipe from here: https://github.com/homuler/MediaPipeUnityPlugin

KEY TAKEAWAYS

Body tracking connects sensing, machine learning, geometry and animation.

Sensing

RGB-D cameras measure depth; RGB pose estimators infer landmarks from visual patterns.

Representation

The body becomes a skeleton graph of landmarks, bones and confidence values.

Application

In VR/AR, tracking must be stable, low-latency and mapped to an avatar rig.

Next step: implement and test a real-time pose tracking exercise with MediaPipe.